

EMULATION OF THE AN/UYK-7
TACTICAL DATA COMPUTER ON THE
BURROUGH'S D-MACHINE

Jerry Michael Haggerty

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

EMULATION OF THE AN/UYK-7
TACTICAL DATA COMPUTER ON THE
BURROUGH'S D-MACHINE

by

Jerry Michael Haggerty
and
John Michael Hartling

December 1976

Thesis Advisor:

S. Jauregui

Approved for public release; distribution unlimited.

7176333

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|--|-----------------------|---|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) Emulation of the AN/UYK-7 Tactical Data Computer on the Burrough's D-Machine | | 5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; December 1976 |
| 7. AUTHOR(s) Jerry Michael Haggerty and John Michael Hartling | | 6. PERFORMING ORG. REPORT NUMBER |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940 | | 8. CONTRACT OR GRANT NUMBER(s) |
| 11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Naval Postgraduate School Monterey, CA 93940 | | 12. REPORT DATE December 1976 |
| | | 13. NUMBER OF PAGES 179 |
| | | 15. SECURITY CLASS. (of this report) Unclassified |
| 16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited. | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |
| 17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) | | |
| 18. SUPPLEMENTARY NOTES | | |
| 19. KEY WORDS (Continue on reverse side if necessary and identify by block number) emulation AN/UYK-7 interpreter microprogramming | | |
| 20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A workable design is presented for emulating the AN/UYK-7 multiprocessing computer system on the Burrough's Interpreter Based System, the D-Machine. The program developed provides for exact execution of the AN/UYK-7 instruction repertoire with the exception of Floating Point, hardware interrupts and IOC Instructions. The design allows for future expansion to incorporate these functions. Input/Output is limited to a card | | |

20. (cont.)

reader, line printer and single disk. Various aspects of Emulation, the D-Machine, and the AN/UYK-7 are discussed and a detailed User's Manual is provided along with recommendations for modifying the design into a full emulation.

Emulation
of the
AN/UYK-7
Tactical Data Computer
on the
Burrough's D-Machine

by

Jerry Michael Haggerty
Lieutenant, United States Navy
B.S., United States Naval Academy, 1970

John Michael Hartling
Lieutenant, United States Navy
B.S., Miami University, Oxford, Ohio, 1969

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
DECEMBER 1976

ABSTRACT

A workable design is presented for emulating the AN/UYK-7 multiprocessing computer system on the Burrough's Interpreter Based System, the D-Machine. The program developed provides for exact execution of the AN/UYK-7 instruction repertoire with the exception of Floating Point, hardware interrupts and IOC Instructions. The design allows for future expansion to incorporate these functions. Input/Output is limited to a card reader, line printer and single disk. Various aspects of Emulation, the D-Machine, and the AN/UYK-7 are discussed and a detailed User's Manual is provided along with recommendations for modifying the design into a full emulation.

CONTENTS

| | |
|---|----|
| ACKNOWLEDGEMENTS..... | 8 |
| I. INTRODUCTION..... | 9 |
| II. EMULATION..... | 12 |
| A. GENERAL CONCEPTS OF MICROPROGRAMMING..... | 13 |
| B. BUILDING AN EMULATOR..... | 17 |
| C. APPLICATIONS..... | 24 |
| III. THE AN/UYK-7..... | 27 |
| A. INTERRUPTS..... | 28 |
| B. CENTRAL PROCESSOR..... | 29 |
| 1. Program Address Register (PAR)..... | 30 |
| 2. Active Status Register (ASR)..... | 31 |
| 3. Base Register (S)..... | 32 |
| 4. Arithmetic Register (A)..... | 32 |
| 5. Index Register (B)..... | 32 |
| C. INSTRUCTION FORMAT..... | 34 |
| D. MODES OF OPERATION..... | 37 |
| IV. BURROUGH'S D-MACHINE..... | 38 |
| A. GENERAL DESCRIPTION..... | 38 |
| 1. Logic Unit (LU)..... | 38 |
| 2. Memory Control Unit (MCU)..... | 42 |
| 3. Control Unit (CU)..... | 43 |
| 4. Microprogram (M-Memory)..... | 44 |
| B. NAVAL POSTGRADUATE SCHOOL CONFIGURATION..... | 47 |

| | | |
|------|---|-----|
| 1. | Description..... | 47 |
| 2. | I/O Interface..... | 49 |
| 3. | Memory Interface..... | 49 |
| C. | INSTRUCTION TIMING..... | 50 |
| D. | LANGUAGES..... | 51 |
| 1. | ALGOL..... | 51 |
| 2. | TRANSLANG..... | 52 |
| V. | PROJECT DESCRIPTION..... | 54 |
| A. | LOADER..... | 56 |
| B. | EMULATION PROGRAM..... | 57 |
| C. | REGISTER MAPPING..... | 67 |
| D. | TIMING..... | 69 |
| VI. | SUMMARY..... | 71 |
| A. | PROBLEMS..... | 71 |
| B. | CONCLUSIONS..... | 73 |
| VII. | RECOMMENDATIONS..... | 75 |
| | APPENDIX A. USER'S MANUAL..... | 78 |
| | APPENDIX B. EMULATOR PROGRAM LISTING..... | 91 |
| | APPENDIX C. LOADER PROGRAM LISTING..... | 139 |
| | APPENDIX D. SAMPLE AN/UYK-7 SORT PROGRAM..... | 154 |
| | APPENDIX E. SAMPLE LOADER OUTPUT LISTING..... | 157 |
| | APPENDIX F. SAMPLE DEBUGGER OUTPUT LISTING..... | 161 |
| | GLOSSARY..... | 167 |

BIBLIOGRAPHY.....175

INITIAL DISTRIBUTION.....178

ACKNOWLEDGEMENTS

We would like to take this opportunity to express our sincere appreciation to the following people for the support they have provided us over the last six months during our pursuit of this thesis. To Professor S. Jauregui, first for sponsoring the thesis, and second for providing funds for research trips to Paoli, Pa. and San Diego, Ca. without which the project could not have been done. To Mr. J. Lynch, Burrough's Advanced Development Organization (ADO), for providing software and hardware engineers on three occasions, and whose personnel graciously answered our questions during almost daily phone calls. To Mr. Carl Benson, Naval Electronics Systems Engineering Center (NESEC), San Diego, for his financial support and personal interest in the project. And last but not least our wives and families who had the patience to see us through this thesis in spite of our 14 hour days away from home, and who soothed our frayed nerves when things did not go right.

I. INTRODUCTION

In its effort to provide the Fleet with officers well versed in all aspects of Computer Science, the Naval Postgraduate School strives to furnish each student in the Computer Science Curriculum with ample opportunities for hands-on operation and programming experience on a variety of computer systems. The systems presently available for this purpose include the IBM 360/67, PDP 11/45, XDS 9300, and several varieties of microcomputers and stand alone graphic systems. Unfortunately, this wide range of available systems and their incorporated programming languages does not include any of the numerous tactical computer systems in use in the Fleet today; moreover, because of budgetary and procurement lead time constraints, it is highly unlikely that the school will have such a system in the near future. To provide the desired systems, the Chief of Naval Education and Training and the Naval Postgraduate School, with the assistance of Burrough's Corporation, have provided a medium through which students may gain experience and perform research on, not one or two Tactical Data Systems but potentially on any particular computer in which the student may express an interest. This medium is the Burrough's Interpreter Based System, a microprogrammable computer, hereafter referred to as the Burrough's D-Machine.

Through microprogramming, the D-Machine can be made to operate exactly like any designated computer of interest.

Through the process of Emulation the authors sought to demonstrate that the D-Machine could be microprogrammed to imitate a selected computer. As the target computer, the authors selected the AN/UYK-7 which is used extensively in Tactical Data Systems and is one of the more complex systems in use today. It was felt that if a realistic emulation of the AN/UYK-7 could be demonstrated, then any lower level, less sophisticated computer could also be successfully emulated.

The goal of this thesis was not a complete AN/UYK-7 emulation, but rather development of the design for the Emulation, and adoption of a sufficient subset of the AN/UYK-7 instruction repertoire to demonstrate exact and efficient execution of AN/UYK-7 programs. The design allows for a complete emulation of all AN/UYK-7 functions and provides for future expansion to a complete emulation including all IOC Functions. Chapters II, III and IV are designed to provide the reader with a general knowledge of Emulation, the AN/UYK-7 and the Burrough's D-Machine. Chapter V describes the methods used for imitating the AN/UYK-7 instructions and its various modes of operation. Chapter V also defines the mapping of the AN/UYK-7 Control Memory Registers and status bits into the Burrough's D-Machine for the purpose of this Emulation. Chapter VI presents some of the problems encountered in the course of this thesis, and

the conclusions drawn by the authors as a result of this Emulation. Chapter VII provides recommendations for expanding this Emulation and suggests possible follow on thesis topics. Appendix A is included as a User's Manual, and contains information on how to use the D-Machine, how to run AN/UYK-7 programs on the Emulator, and how to use special features of the microprogramming language TRANSLANG which were not documented in the TRANSLANG Programmer's Manual. The program listings of the Loader and Emulator, written in association with this thesis and used to demonstrate the feasibility of the design by running AN/UYK-7 Programs, are included as Appendix B and C. A sort routine written in AN/UYK-7 machine language, and used to demonstrate the Emulator's capability is provided in Appendix D. The output of the sort program and the optional Loader functions of assembler and debugger are presented as Appendix E and F.

II. EMULATION

Modern computer systems are generally composed of five basic units: input, output, memory, arithmetic/logic, and control. The instruction execution and communications among these units are usually well understood with the exception of the control unit which is, typically, understood only by the system engineer. The control unit generates the signals necessary for the information flow and timing of the system. In conventional computers this is done through the use of flip-flops (e.g. registers and counters) and gates designed in a relatively ad hoc manner. Microprogramming the control unit has been proposed as an orderly alternative to this ad hoc design procedure where the hardware of the control section is replaced by a "microprogram control" unit.

Microprogramming is therefore a technique for implementing the control function of a digital computer using programmable control signals stored in a separate, word-organized memory, called control store. If the control store were a programmable memory, then the system architecture could be modified to optimize processing of each task to be performed; moreover, it could be altered completely to resemble the architecture of another, entirely different machine. The microprogramming of the control store of a computer system to alter the basic architecture enabling one

machine (the host machine) to execute machine language programs intended for another machine (the target machine) is defined as Emulation.

The remainder of this chapter is divided into three sections: general information on microprogramming, construction of an emulator and applications of emulation.

A. GENERAL CONCEPTS OF MICROPROGRAMMING

Since the cost of software has become a major portion of the overall cost of computer systems today, more and more manufacturers and users are turning to microprogramming the control section of their computer to tailor the machine to individual applications, thereby making computer programming more efficient.

The main function of the control section is to specify the conditions under which sets of gates are to be opened. In conventional machines this can lead to a very complex, hardwired system, especially if the instruction set is extensive. A relatively simple field change, addition or alteration, may require a major modification of this complex system. This is not true in a computer that uses a microprogrammable control store. The complex hardwired structure is replaced by microinstructions stored in the control store, one microinstruction per word, which tell the system which arithmetic and logic operations to perform by specifying which gates to enable/disable. In this way,

control of the computer is removed from the hardware and placed under the control of the microprogrammer. This places a heavy burden on the programmer since he must become intimately familiar with the innermost workings of the machine; however, it allows him to model the machine to his specific programming needs. This gives the microprogrammer extreme flexibility in his applications on the computer.

There are two types of microinstructions: vertical and horizontal. Vertical microinstructions usually control one operation; and the address of the successor microinstruction is implicitly the current address plus one, unless the current microoperation causes a branch. In horizontally microprogrammed machines each control bit of a microinstruction is assigned to a specific gate or set of gates. This means that one microinstruction can control multiple operations. On the other hand, vertical microinstructions are divided into separate fields, each of which are then decoded to enable/disable specific gates or sets of gates; therefore, horizontal microinstructions require less decoding and provide more flexibility. However, the advantage of vertical microinstructions is that they generally require shorter control words considerably reducing the overall cost of control micromemory. The length of control words vary from 20 bits (vertical) to in excess of 120 bits (horizontal); the average microinstruction is 50-80 bits. The D-Machine, the host machine for this thesis, utilizes a combination of

vertical (Type II) and horizontal (Type I) microinstructions in the form of a 56-bit word.

Since the execution time of one horizontal microinstruction is essentially the same as that of one vertical microinstruction, the multiple operations performed during one horizontal instruction is a desirable feature. This parallelism can drastically reduce the size of a microprogram, and dramatically decrease execution times. These reductions can only be realized, however, if the programmer can recognize concurrent actions and optimally group them into one instruction. A great deal of work has been done toward including the efficient use of available resources into a compiler for a high-level microprogramming language, but with little success. Either the probability of making the determination is low, or the cost of the optimization is too high. Much additional research is required in the area of recognizing and combining potentially concurrent actions.

Different microinstructions specify different microoperations to be performed, and depending on the nature of the microoperations, some phase of one instruction may overlap a phase of a subsequent instruction. Instruction timing is, therefore, an important factor in the overall efficiency of microprograms in which microoperations are not mutually exclusive. A brief description of the timing considerations that must be made in microprogramming the Burrough's D-Machine is given in Chapter IV.

Many main-frame manufacturers hesitate to allow the novice programmer access to the innermost workings of their computer (e.g. registers, data paths, and gates). One method of prohibiting this access is to use read-only micromemory with the microprograms being provided by the computer manufacturer. This helps preserve the designed identity of the computer but does not provide for its optimum use. The programmer who is given the capability of modifying the logical design of the machine to his specific programming needs will find his programming tasks greatly simplified. For example, he could implement new instructions, such as floating point, which were not designed into the original machine.

In summary, microprogramming is becoming more widely used for the following reasons: 1) the flexibility and growth potential of microprogrammable machines (especially in the area of emulation), 2) increasing software costs and current semi-conductor technology favor microprogram design, and 3) user-oriented instructions can be designed into machines through microprogramming.

B. BUILDING AN EMULATOR

Emulation describes a process through which the hardware components of one machine (host) are made to "appear" to assume the specific characteristics of the hardware of another machine (target). This provides the host machine with the capability of executing machine language instructions written for the target machine. Simulation is a software process that provides the same capability for program execution; however, simulation involves interpretive execution by a high-level language program. Emulation differs in that an emulator performs its interpretive execution through the hardware. Simulations usually perform within a factor of 100-200 times real-time. Emulation is generally within a factor of 10, but often can be tuned to approach real-time. For this reason emulations are usually more cost-effective than simulations. The following paragraphs describe the process involved in building an emulator.

There are two major approaches to emulating a target machine: 1) the hardware and firmware (the physical realization of a microprogram) can be used in conjunction with a software simulator (software alone would be pure simulation), or 2) the hardware and firmware can execute with no software support.

Depending on the percentage of software utilized, a software-assisted emulation may be somewhat slower than a

firmware emulation. Typically, the programmer mentally develops a software simulation of the target machine. He then decides which sequences of instructions are most frequently used, and which are the hardest to simulate. These become candidates for microprogramming. Frequently a single new instruction can be microprogrammed to replace repetitive sequences of the same high-level instructions, thus speeding up the emulation. With this software-firmware approach the user must decide the point at which he must stop substituting microcode for software routines. This is generally dependent on the amount of control storage available, or the cost-performance criteria with which he is working. An example of a software-hardware approach to emulation is the IBM 7000 series emulators used with the IBM System/360 Model 65.

The Burrough's D-Machine provides the capability of emulation using the software-hardware method. A simulation is written in the high-level language ALGOL. ALGOL on the D-Machine allows the user to define a new operator which, when called in the simulation program, branches to a preprogrammed series of microinstructions; executes the microinstructions; and returns to the ALGOL program. This gives the user the ability to execute frequently used routines (such as instruction fetch) from micromemory which has a faster cycle time than that of main memory where the ALGOL simulator resides. An advantage to this approach is that the system does not have to be regenerated after different emulators

are run since, in essence, only an ALGOL program has been executed. However, as previously discussed, this software emulator is slower than a firmware-controlled emulator.

The authors used the hardware-firmware approach in their emulation of the AN/UYK-7. This method will be developed in more detail along with suggested hardware additions which could enhance the emulation. Control of the system resides entirely in the firmware, which means that the emulator, once loaded, dedicates the machine to that emulation, and native mode programs can no longer execute until system regeneration. This unproductive overhead of loading and reloading the emulator and the native mode machine is one drawback. An additional drawback is the larger micromemory required since the entire emulation is microprogrammed. However, the emulator executes exclusively through firmware making this approach to emulation significantly faster. An example of a firmware-controlled emulator is the IBM 1401 emulator on the System/360 Model 30.

Although systems such as the D-Machine use horizontal microinstructions to enable parallel or concurrent operations, the microprogram can only perform one function of the target machine at a time. That is, if the target machine performs parallel operations such as executing one instruction while simultaneously fetching the next instruction, the microprogram can only emulate one of these functions at a time. It must execute the current instruction and then fetch the next instruction. For this reason, emulation is

usually slower than the real time performance of the target machine even though it may have a faster memory cycle time.

In order to emulate a target machine, the microprogrammer must first understand three areas: 1) the host machine, 2) the target machine and 3) the microlanguage. Second, the registers, counters, and accumulators of the target machine must be mapped into the host machine. This is often done in two steps - depending on the number of available registers in the host machine. The registers of the target machine most frequently used should be mapped directly to registers in the host where possible. The remaining registers of the emulated machine are mapped into the host's main memory. As many of the target machine's registers as possible should be mapped directly, so that fewer memory references will be required by the emulator. For example, if the emulator were executing a Load Accumulator instruction and the specified accumulator had been mapped into one of the host's registers, then only one microinstruction is needed to emulate the load instruction. However, if the accumulator had been mapped to a memory location, then it would take several microinstructions to emulate the entire operation - i.e. setup the store address, store the value to be loaded into the write register, and perform a write to main memory. Not only does this require additional instructions, but it also includes a main memory access, whose cycle time is slower than micromemory's cycle time. Some instructions may require two main memory cycles, such as an ADD where the

mapped accumulator must be read, added to and then written out. This demonstrates the necessity of mapping as many registers as possible to the host machine's hardware. Sel-dom, however, can this mapping be accomplished without some use of main memory; therefore, the microprogrammer must select some portion of main memory as a privileged area for register mapping. The authors reserved the first 1024 words of main memory in the emulation of the AN/UYK-7, for register and buffer mapping.

The next step in the emulation is to assume that a user's object program is resident in main memory. The emulator must then fetch, decode and execute the program, instruction by instruction. The microprogrammer must decide which emulation functions should be written as subroutines, and which functions should be written in line. The most frequently referenced subroutine in any emulation is the fetch routine since it is executed for every instruction. The normal steps involved in this routine are:

1. Determine the address of the next instruction as indicated by the program address register.
2. Read the instruction from the main memory address calculated in step 1. This is the current instruction to be emulated.
3. Decode the instruction into its designator fields.
4. Calculate the operand address. (The AN/UYK-7 uses a displacement plus an index register plus a base register).
5. Read the operand from main memory.

6. Perform indirection if allowed and indicated.
7. Some instruction sets, AN/UYK-7 for example, allow whole or half-words. Check if current instruction is half-word.
8. Some instruction sets operate on partial word operands. If so determine if quarter, half or full-word operand is to be used.

These steps must be executed for each instruction fetched from the user's program. Depending on the particular instruction being emulated, some of these steps may be omitted (step 8 is only performed for Format I instructions) but the majority are generally applicable. This demonstrates the overhead involved in emulation prior to branching to the microroutine that does the actual instruction execution.

Some hardware additions that can facilitate emulation are a fast shifter and a field select unit (FSU). The shifter simplifies variable-length byte extraction, a common emulation process. For instance, in the D-Machine it takes the same amount of time to shift a word by 1 bit as it does for any shift up to 31 bits. The FSU allows the testing of selected fields without requiring a cycle to go through the adder (the D-Machine does not have this feature). This relieves the programmer of masking and shifting fields prior to testing them. These two features are the most common and cost-effective additions to the hardware.

One approach to emulation that has recently come into use, but which is not that widely used or understood, involves using a combination of hardware, software and operating system. IBM is experimenting in this area, and has established the following design characteristics for the System/370 emulators:

1. Emulators must be integrated with the operating system and run as a problem program. This eliminates the overhead of loading and reloading the emulator and the native mode machines.
2. Allow multiprogramming of emulators as well as a mix of emulators and native mode programs.
3. All programs including emulators must be interruptible.

With such a system IBM felt they could give the user an improved, more efficient environment from which to operate, either in the emulator or native mode. The authors feel this should be a primary field for future applications.

The final item to be discussed in the Section is emulation of Input/Output operations. Most sources agree that emulation of I/O is as difficult, if not more so, than the implementation of the central processor's instruction set. This is true primarily because once the basic routines needed for instruction and operand fetch are programmed, the actual instruction executions are fairly easy to microprogram. This is not necessarily true with I/O - buffers must be set up; data conversion possibly performed; and perhaps the hardest point, emulating circumstances in which results are

not yet known. For instance, if the I/O is to search for a specified record, then what happens if the key cannot be found? There are many such checks or error conditions that make I/O difficult to emulate. Another major problem is emulating interrupt handling during I/O. Since emulation of I/O is essentially a separate topic and since the authors did not implement emulation of the AN/UYK-7 Input/Output instruction set as part of the thesis, I/O emulation will not be described in any further detail. However, references 5, 12, 15, 21, and 22 provide excellent material on this topic.

C. APPLICATIONS

This section is designed to familiarize the reader with some of the applicable areas for emulation usage. It is by no means an exhaustive study of this topic, nor is it intended to be so.

Emulation was first used in, and is still the primary application for, converting from second generation computing systems to third generation systems. Normally, the process is costly and dangerous in the aspect of converting programs running on the existing system to programs capable of executing on the new machine. There are several methods other than emulation capable of performing this task, none of which are completely satisfactory: simulation is slow; automated translation is unproven; and instruction reprogramming is costly and cumbersome. The advantage of emulation is that if the old machine could be emulated on

the new machine allowing direct execution of all old programs, then the old programs could be converted at leisure, if desired and justified. For example, reprogramming might not be cost effective or justified for a program with an expected life span of six months. In such circumstances continued emulation of the program is the solution. However, the situation might be different if the program had an expected life span of three years.

Another major application for emulation is the design, development, and testing of a newly conceived system on an existing system. The target machine does not necessarily have to be an existing computer system, and the user could experiment with the design of any machine he desires. In this way a richer instruction set can be achieved; moreover, software and diagnostic routines could be developed for the new machine prior to the machine even being produced or marketed. IBM used this method to some extent by emulating System/370 on System/360 prior to building System/370. This application tends to decrease the time from initial construction to marketing because software, such as the operating system, can be tested and debugged concurrently.

A third application, and one seen as a major trend for the future, is the ability of the user to adapt the computer to his individual needs. Modern-day computers are built as powerful, general-purpose machines designed to operate effectively over a large field of problems. In doing so, however, the machine cannot optimize execution of any one

particular application. Emulation gives the user this ability. The user can either operate from an existing emulator, or he can tune the emulator to his individual application through microprogramming. One user may want a machine oriented toward string operations, while another user may be doing extensive scientific calculations. If each user could execute from a separate emulator tuned to his individual application, then each would be more efficient and productive. In addition the computer would probably be easier for the user to program. This is the area where multiprogramming of emulators is projected to be of the most use.

There are several other areas of emulation usage which were not discussed; such as the academic environment for research purposes (the reason for this thesis), and tuning of existing software. However, as previously stated, this section was designed only to give the reader a flavor of the possible uses of emulation and perhaps stimulate his own ideas for possible applications. For additional and more detailed material see references 2, 15 through 18, and 20 through 24.

III. THE AN/UYK-7

This chapter is intended to introduce the reader to those operational characteristics and capabilities of the AN/UYK-7 computer most pertinent to the Emulation. This will assist the reader in understanding the design and mappings used in this Emulation and presented in subsequent chapters. This chapter is not intended to be a technical manual and should not be used for that purpose. The reader should consult references 26 and 27 for more detailed information.

The AN/UYK-7 is a highly reliable, ruggedized, multiprocessor system designed and manufactured by the Univac Division of Sperry Rand Corporation. Built to military specifications (MIL-E-16400), it is utilized extensively throughout the United States Navy in Tactical Data Processing applications [26]. Rapid data transfer rates are provided during communications between external devices, internal random-access memory, and the central processor. The system is capable of average command execution times of 1.5 microseconds, and an input data transfer rate of over one million 32-bit words per second. The AN/UYK-7 maintains two completely separate sets of index, arithmetic and relative address (Base) registers, for use during Task and Executive states. In addition, a set of 18 privileged instructions

with special characteristics for executive control are reserved for the Interrupt State. These features along with the parallelism of AN/UYK-7 field and register references, make it an extremely difficult system to emulate in real-time.

The AN/UYK-7 was designed to operate as either a single or multiple processor system with up to 256K, 32-bit words of random access memory. This allows on-line access to 1024K bytes through an instruction feature which permits whole-word, half-word, or quarter-word memory references. This Emulation assumed an AN/UYK-7 configuration consisting of a single processor, monoprogramming, with one 64K, 32-bit word memory module. Partial word references were fully emulated providing random access to 256K bytes.

A. INTERRUPTS

The AN/UYK-7 processes data in real-time in response to a highly prioritized interrupt system with decision-making properties. This priority system allows the central processor to select that program routine which is necessary to respond to the interrupt requiring the most urgent attention. Since the Emulator was designed for a monoprogramming environment using asynchronous I/O, the interrupt features of the AN/UYK-7 were not fully implemented. Several types of interrupts, such as Program Faults (Illegal Instructions), were implemented and tested. All interrupt handling flags, lockouts and registers were mapped into the Emulator.

This will facilitate full implementation of interrupt features - should a decision be made to incorporate multiprogramming or synchronous I/O in the course of expansion to a full Emulation.

B. CENTRAL PROCESSOR

The AN/UYK-7 central processor contains all the control, arithmetic and timing circuitry required for processing alphanumeric data and for executive functions. The central processor operates in two different modes or states: the Interrupt State executes executive-type functions, and the Task State processes user programs. A group of 16 privileged instructions and a separate set of arithmetic, index and base registers are reserved for the exclusive use of the processor while in the Interrupt State. These features greatly enhance the AN/UYK-7's multiprocessing and multiprogramming capabilities. These special instructions and registers were mapped into the Emulator; however, they were not fully implemented because of the monoprogramming environment in which the Emulator was to be run.

The AN/UYK-7 maintains a central processor control memory (CMR) consisting of 82 integrated-circuit, random-access registers of varying sizes appropriate to their function (e.g. A-registers 32-bits, B-registers 20-bits). The various registers are grouped into stacks according to their use, addressing, and relative size. In the Emulator, these registers were mapped into the D-Machine's main memory

starting at address 0000, with address assignments corresponding exactly to that of the AN/UYK-7. The notable exception was that CMR addresses designated as "Unassigned" in the AN/UYK-7 were used as temporary storage registers in the Emulator. Register mapping is discussed in more detail in Chapter V and presented in tabular form in Figure V-2.

The AN/UYK-7 registers of particular interest to the user are: the Program Address Register (PAR); the Active Status Register (ASR); and the Base, Index and Arithmetic Register groups. These are the only registers which are either directly addressable or accessible to the programmer.

1. Program Address Register (PAR)

The PAR is a 20-bit register used for addressing the next instruction to be fetched from memory for execution. It consists of a 16-bit displacement value and a 3-bit Base Register reference. The effective instruction address is formed by the addition of the displacement and the contents of the selected Base register. The PAR displacement value is incremented by one word at the completion of each instruction cycle with the exception of JUMP instructions. These instructions cause replacement of the PAR by the "s" and "y" operand fields of the JUMP, thus providing for out-of-sequence instruction execution. The PAR mapping in the Emulator is not composed of two separate fields, but rather as the calculated effective address. This greatly simplifies and expedites instruction references; however, it

causes some difficulty with specific instructions which require access to the two separate fields of the PAR. When encountered, this problem was overcome by dividing the PAR value by 8K. Since the Base registers were preinitialized in 8K increments, starting at 1024, the results of the division yielded a quotient equivalent to the Base register assignment, and a remainder equal to the displacement.

2. Active Status Register (ASR)

The ASR is a 23-bit register used to indicate and control the status of various operations in the central processor. Individual bits of the register are assigned special functions, and when set indicate that a particular status exists and that the processor should be controlled accordingly. Individual bits are set/cleared as the result of either an instruction execution or direct processor intervention. The bits of particular interest to the programmer are the arithmetic bits (equal, greater than or equal, overflow, and limits) which are set as the result of arithmetic functions, and may be interrogated by specific instructions, such as conditional jumps. Based on the condition of the bit tested these conditional instructions may cause a change in the program sequence. In the Emulator the ASR bits most frequently referenced by the programmer are mapped into a D-Machine internal register together with the PAR. This considerably reduces main memory references, and reduces Emulator execution time. The ASR mapping is discussed in detail in Chapter V.

3. Base Register (S)

There are two sets (Interrupt, Task) of 18-bit S-Registers, numbered 0-7. These registers, used in final Y-Operand and instruction address calculations, are necessary in a multiprogramming and multiprocessing environment. In spite of the monoprogramming environment of the Emulator, the registers were mapped and used as designed; however, they were preinitialized by the Loader at 8K boundaries starting at address 1024. This provided the Emulator access to 64K of main memory in increments of 8K pages.

4. Arithmetic Register (A)

There are two sets (Interrupt, Task) of 32-bit A-Registers, numbered 0-7. They are used extensively throughout the instruction set to hold one or more of the operand - inputs to, or results of arithmetic functions. The A-Registers also serve as the main interface between the central processor and main memory. These registers are directly addressable by the programmer.

5. Index Register (B)

There are two sets (Interrupt, Task) of 20-bit B-Registers, numbered 1-7. These registers can be used as counters, or they can be employed in a special addressing technique called Indexing. During normal Y-Operand address calculations, the designated B-Register is added to the y-displacement and a specific S-Register to form an effective

address. This address is then considered to have been indexed by the B-Register value. Indexing provides the programmer with the option of sequentially referencing memory by merely incrementing the index during each pass through a loop. Reference to any B-Register other than B(0) implies that indexing is to take place during the Y-Operand address calculation. A reference to B(0) is treated as an index of zero since B(0) is not a valid register. In the Emulator, the contents of B(0) is always zero, and references to B(0) are conveniently treated the same as references to B(1) thru B(7). In the AN/UYK-7, the B-Registers have the same internal structure as the PAR; that is, they consist of two fields (a Base register, and a displacement). In the Emulator, B-Registers are treated as consisting of one field which can be separated into its two respective fields by division by 8K, as previously described in the PAR discussion.

In order to minimize access time and field decoding, all CMR registers were treated as 32-bit registers in the Emulator. This caused no problems with the exception of the PAR and Index Registers, which were resolved as previously discussed. The A, S, and B-Registers are directly addressable through the Load/Store CMR Instructions of the AN/UYK-7. They are incidentally addressable through references in specific fields of the different Format instructions reserved for that purpose.

C. INSTRUCTION FORMAT

Instructions of the central processor appear in five different formats according to their operational characteristics, as presented in Figure III-1. Formats I, II, and III occupy a full, 32-bit computer word; Formats IV-A and IV-B each occupy a half computer word. Two half-word instructions can be stored in one memory location. When a half-word instruction in the upper half of the computer word is executed, the processor sets bit 15 of the ASR; if a whole-word or lower half-word is executed the bit is cleared. In the Emulator, this bit determines whether the subroutine IFETCH or HALFFETCH is executed. IFETCH reads the next 32-bit instruction from memory. HALFFETCH shifts the lower half-word of the current instruction into the upper halfword position for execution.

Each of the five formats divide the instruction into different fields. Each field except "y" (the constant or address field) has a particular function in controlling the various internal enables and commands required for proper execution of the instruction. Some fields define the use, modification or application of the y-field to secure the desired operand; others select an accumulator, index, or base register; others select an IOC, define a sub-function code, or combine to form a special interpretation defined by the instruction. The AN/UYK-7 maintains a repertoire of 132 central processor instructions whose specific functions are defined in detail in reference 26.

| | | | | | | | |
|------------|--------|-------|-------|-------|----|-------|--------|
| Bit No. | 31--26 | 25-23 | 22-20 | 19-17 | 16 | 15-13 | 12---0 |
| FORMAT I | f | a | k | b | i | s | y |
| FORMAT II | f | a | f2 | b | i | s | y |
| FORMAT III | f | a | f3/k | b | i | s | y |

| | | | | | |
|-------------|--------|-------|-------|-------|----|
| Bit No. | 31--26 | 25-23 | 22-20 | 19-17 | 16 |
| Bit No. | 15--10 | 9-7 | 6-4 | 3-1 | 0 |
| FORMAT IV A | f | a | f4 | b | i |
| FORMAT IV B | f | a | m | | |

INDIRECT CONTROL WORD FORMATS (Indirect Addressing)

| | | | | | | | |
|---------|-------|-------|--------|-------|----|-------|--------|
| Bit No. | 31-30 | 29-25 | 24-20 | 19-17 | 16 | 15-13 | 12---0 |
| | c | w | p | b | i | s | y |
| | c | c1 | unused | b | i | d | |

Elements of the word are interpreted as follows:

| FIELD | BASIC DEFINITION |
|-------|--|
| f | 6-bit function code |
| f2 | 3-bit subfunction code |
| f3 | 2-bit subfunction code |
| f4 | 3-bit subfunction code |
| a | 3-bit accumulator register designator |
| k | 3-bit operand interpretation designator |
| m | 6-bit shift count designator |
| b | 3-bit index register designator |
| i | 1-bit indirect addressing designator |
| s | 3-bit base designator |
| y | 13-bit address displacement/operand designator |
| c | 2-bit control designator |
| c1 | 1-bit indirect subfunction designator |
| w | 6-bit character length designator |
| p | 5-bit position indicator for character length |
| d | 16-bit address displacement |

INSTRUCTION AND INDIRECT ADDRESS WORD FORMATS [27]

FIGURE III-1

Of particular note to the reader is the K-Field of the Format I instructions. The value of K designates whether the operand to be fetched/stored is a whole, half, or quarter-word operand, as depicted in Figure III-2. If the value of K implies a partial-word operation, then it also determines which portion of the 32-bit word is to be accessed. That is, a K-value designating a quarter-word (byte) transfer can also specify that the byte is located in the upper, lower, or one of the intermediate two bytes of the 4-byte, 32-bit operand. It is this function that makes the AN/UYK-7 an extremely powerful word processing machine. This function was fully implemented and tested in the Emulator.

| K | READ MEMORY to ARITHMETIC | STORE ARITHMETIC to MEMORY |
|---|------------------------------|---|
| 0 | sy(SE)+B(b) -> A15-0(SE) | NOT USED |
| 1 | Y15-0 -> A15-0 (SE) | A15-0 -> Y15-0; Y31-16 unchg |
| 2 | Y31-16 -> A15-0 (SE) | A15-0 -> Y31-16; Y15-0 unchg |
| 3 | Y31-0 -> A31-0 | A31-0 -> Y31-0 |
| 4 | Y7-0 -> A7-0 (ZE) | A7-0 -> Y7-0; Y31-8 unchg |
| 5 | Y15-8 -> A7-0 (ZE) | A7-0 -> Y15-8; Y31-16 unchg Y7-0 unchg |
| 6 | Y23-16 -> A7-0 (ZE) | A7-0 -> Y23-16; Y31-24 unchg Y15-0 unchg |
| 7 | Y31-24 -> A7-0 (ZE) | A7-0 -> Y31-24; Y23-0 unchg |
| SE--sign extended; ZE--zero extended A is the ACCUMULATOR specified by the a-field | | |

FORMAT I INSTRUCTION K-FIELD INTERPRETATION [27]

FIGURE III-2

D. MODES OF OPERATION

There are several different modes of operation of the AN/UYK-7; most of them are concerned with different addressing techniques. Specifically, they are: normal mode, index mode, repeat mode, and indirection. In the normal mode, the Y-Operand address is calculated as the sum of: the y-offset, a base register and an index register, or $Y = y + B(b) + S(s)$. There are two exceptions to this general formula. First, if the K-field of Format I instructions is a zero then $Y = sy + B(b)$, where "sy" is the concatenation of the s-field and y-field of the instruction. Second, for all instructions regardless of Format, if the B-field is zero then the B(b) value used in calculating the Y-Operand address is always zero. There are several additional addressing techniques employed during the repeat and indirect modes of operation. They are described in detail in Chapter V. These ad hoc addressing techniques employed by the AN/UYK-7 greatly enhance its capabilities but were extremely difficult functions to emulate.

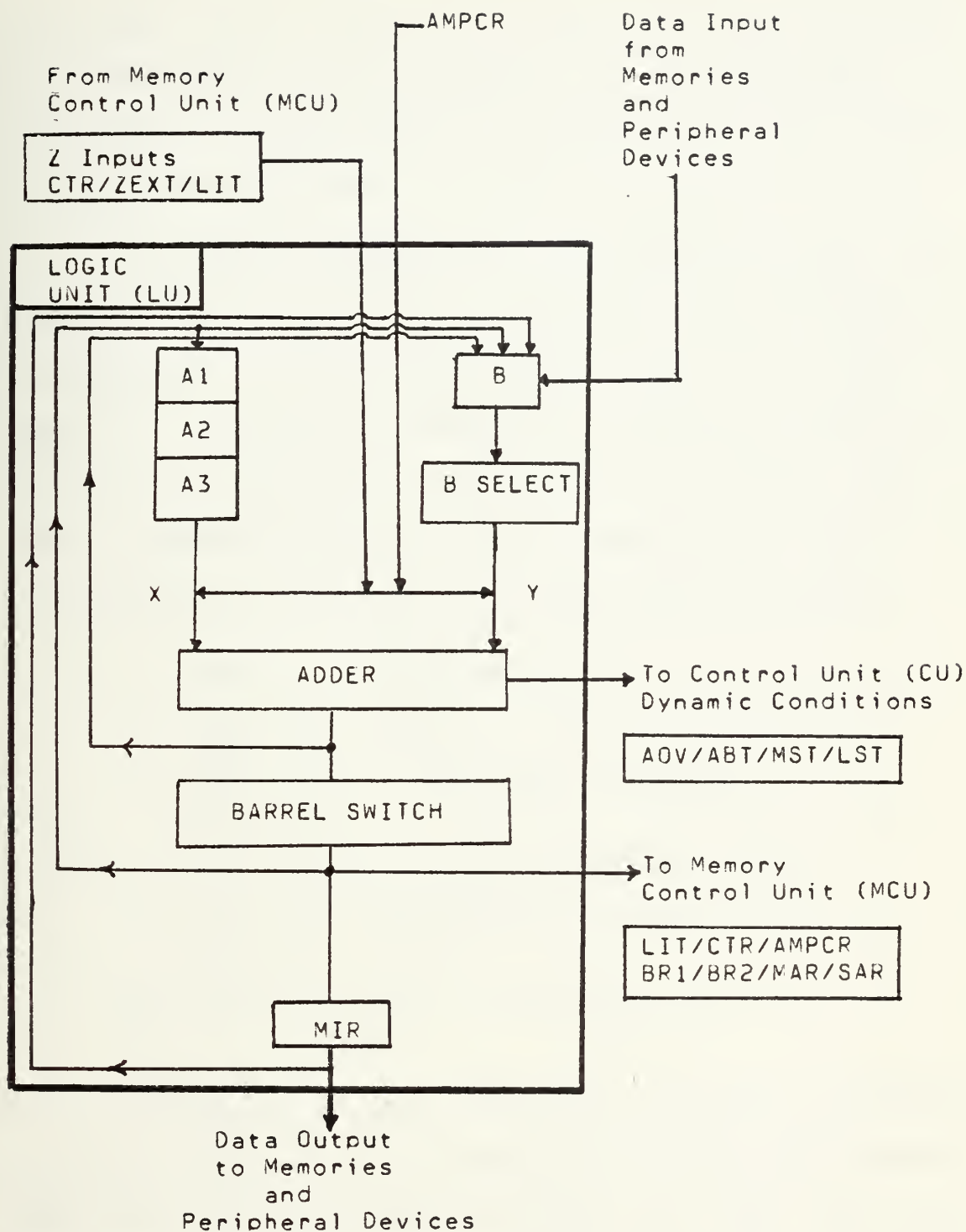
IV. BURROUGH'S D-MACHINE

A. GENERAL DESCRIPTION

An Interpreter Based System is a digital processing concept developed by Burrough's Corporation, that utilizes hardware building blocks which can be tailored through microprogramming to perform as a variety of general purpose or special purpose general processors [7, 8, 24]. The basic Interpreter, also referred to as the D-Machine, is the primary building block of this unique system. It is composed of five functional modules each of which can be modified to manipulate an 8 to 64-bit word format in increments of 8 bits. In the Naval Postgraduate School configuration, two of these modules, the Nanomemory and the Micromemory, are combined into one. The register sizes given in the following descriptions of the modules are specific to this configuration which is a 32-bit D-Machine.

1. Logic Unit (LU)

The Logic Unit (LU), presented in Figure IV-1, contains all the registers, the Barrel Switch and the Adder which are available to the microprogrammer for manipulating data fields during the process of emulating an instruction. A description of each of these registers and its functions follows [8,24].



LOGIC UNIT (LU) BLOCK DIAGRAM

FIGURE IV-1

The 32-bit registers A1, A2, and A3 are functionally identical. They temporarily store data within the Interpreter and serve as the primary (X) input to the adder. Any A-Register can be designated as a destination for the output from the Barrel Switch.

The 32-bit B-Register is the primary external input interface (from the Switch Interlock). It also serves as the secondary (Y) input to the Adder, and can receive the output of the Adder, in addition to the Barrel Switch which is the normal source for the results of arithmetic calculations. The B-Register can be the destination of any of the following during execution of any one microinstruction:

- a. The Barrel Switch output.
- b. The Adder output.
- c. The external data from the Switch Interlock.
- d. The Memory Information Register (MIR).
- e. The carry complements of four-bit or eight-bit groups.
- f. The Barrel Switch output ORed with b, c, or d above.

The output of the B-Register has true/complement selection gates which are controlled in three separate sections: the most significant bit (MSB), the least significant bit (LSB), and all the remaining internal bits. Each of these parts is controlled independently, and may be either all one's, all zero's, the true contents or the one's complement of the respective bits of the B-Register.

The 32-bit Memory Information Register (MIR) buffers the information which is to be written to main system S-Memory, or to a peripheral device. It is loaded from the Barrel Switch output and its output is directed either to the Switch Interlock or the B-Register.

The 32-bit Adder in the LU is a modified version of a straightforward carry look-ahead adder. Inputs to the Adder are from selection gates which allow various combinations of the A, B, and Z inputs. The A input is from the A-Register output selection gates and the B input is from the B-Register true/complement selection gates. The Z input is an external input to the LU and can be:

- a. The output of the counter in the Memory Control Unit (MCU) into the most significant eight bits with all other bits being zeros.
- b. The output of the literal register in the MCU into the least significant eight bits with all other bits being zeros.
- c. An optional input into the middle bits with the most and least significant bytes being zeros.
- d. All zeros.

There are always two inputs to the Adder referred to as the X-input and Y-input. An X-input may have A, Z, or zero as its source. A Y-input may have B, Z, or one as its source. An unspecified X-input is always assumed to be zero. Using various combinations of the possible inputs to the selection gates, any two of the three (A, B, or Z) inputs can be added together, or can be added together with an

additional one added to the least significant bit. In addition, all binary Boolean operations can be performed between any two inputs.

The 32-bit Barrel Switch is a matrix of gates that shifts a parallel input data word from the Adder, any number of places to the left or right, either end-off or end-around. The output of the Barrel Switch may be gated to one or more of the following simultaneously:

- a. The A-Registers (A1, A2, A3).
- b. The B-Register.
- c. Memory Information Register (MIR).
- d. Least significant 16 bits of MCU registers (BR1, BR2, MAR, AMPCR, CTR).
- e. Least significant 5 bits to the Control Unit (CU) for the shift amount register (SAR).

2. Memory Control Unit (MCU)

One MCU is required for an Interpreter to have access to 64K of main memory [8, 24]. The addition of a second MCU is possible, thus expanding on-line memory access to 128K. The MCU has three major sections:

- a. The microprogram address section contains a microprogram count register (MPCR), the 12-bit alternate microprogram count register (AMPCR), the incrementer, the microprogram address controls register, and their associated control logic. This section is used to address the Microprogram Memory (MPM) for the sequencing of microinstructions. The AMPCR contents may be used as a Y-input to the Adder.

- b. The memory/device address section contains the 8-bit memory address register (MAR), the two 16-bit base registers BR1 and BR2, the output selection gates, and their associated control logic.
- c. The Z register section contains registers which are the Z inputs to the LU Adder: a loadable counter (CTR), the literal register (LIT), selection gates for the loadable counter and their associated control logic.

3. Control Unit (CU)

The Control Unit (CU) has five major sections: the shift amount register (SAR), the condition register (COND), part of the control register (CR), the MPM content decoder and the clock control [8, 24].

The functions of the SAR and its associated logic are:

- a. To load shift amounts into the SAR for use in (0 to 32-bit) shifting operations.
- b. To generate the required controls for the Barrel Switch to perform the shift operation indicated by the current microinstruction.
- c. To generate the "word length complement" of the SAR contents, where the "complement" is defined as the amount that will restore the bits of a word to their original position after an end-around shift of N followed by an end-around shift by the "complement" of N.

The control register (CR) is a 56-bit register that stores all those control signals from the current microinstruction which are not used in phase I. The CR is divided into the phase III controls and the MPAD controls.

The condition register (COND) section of the CU performs four major functions:

- a. Stores 12 resettable condition bits in the condition register. The 12 bits of the condition register are used as interrupts, error indicators, status indicators, and lockout indicators.
- b. Selects 1 of 16 condition bits; 12 from the condition register and 4 dynamically generated during the present clock time in the Logic Unit for use in performing conditional operations.
- c. Decodes bits from memory for resetting, setting or requesting the setting of certain bits in the condition register.
- d. Resolves priority among Interpreters in the setting of global condition (GC) bits which provide a facility for inter-Interpreter lockout control.

4. Microprogram (M-Memory)

The Micromemory (M-Memory), also known as the Nanomemory (N-Memory) in the Naval Postgraduate School configuration, is a programmable control store memory which contains the user supplied microprograms in the form of microinstructions. Each microinstruction consists of a 56-bit nanoinstruction and provides the gating for functioning of the previously discussed LU, MCU, and CU modules. Micromemory consists of two 4K, 56-bit, modules allowing the

programmer access to approximately 8K of control store. The sequencing of microprogram instructions is controlled by the following procedure: the Nanomemory provides information to the condition testing logic indicating which condition is to be tested. The condition testing logic provides a TRUE/FALSE signal to the successor logic which selects between the three TRUE and three FALSE successor bits. These three successor bits provide eight possible successor command combinations which are listed below with their associated interpretations:

- | | |
|---------|--|
| a. WAIT | Repeat the current instruction |
| b. STEP | Step to the next Instruction |
| c. SKIP | Skip the next instruction |
| d. JUMP | Jump to address in AMPCR |
| e. RETN | Return from a micro subroutine |
| f. CALL | Call a micro subroutine |
| g. SAVE | Save the address of the head of a loop. |
| h. EXEC | Execute one instruction out of sequence. |

The particular successor command specified then provides the controls needed in the selection (MPCR/AMPCR) and incrementing logic to generate the next MPM address. Except for the EXEC command the MPCR is loaded with the MPM address.

For a more thorough description and discussion of these five modules, reference 10 provides an outstanding

diagrammatic breakdown of the Interpreter's internal hardware configuration and operation. Reference 7 discusses the interface of various Interpreter, peripheral, and memory configurations that have been tested by the Advance Development Organization of Burrough's Defense Space and Special Systems Group.

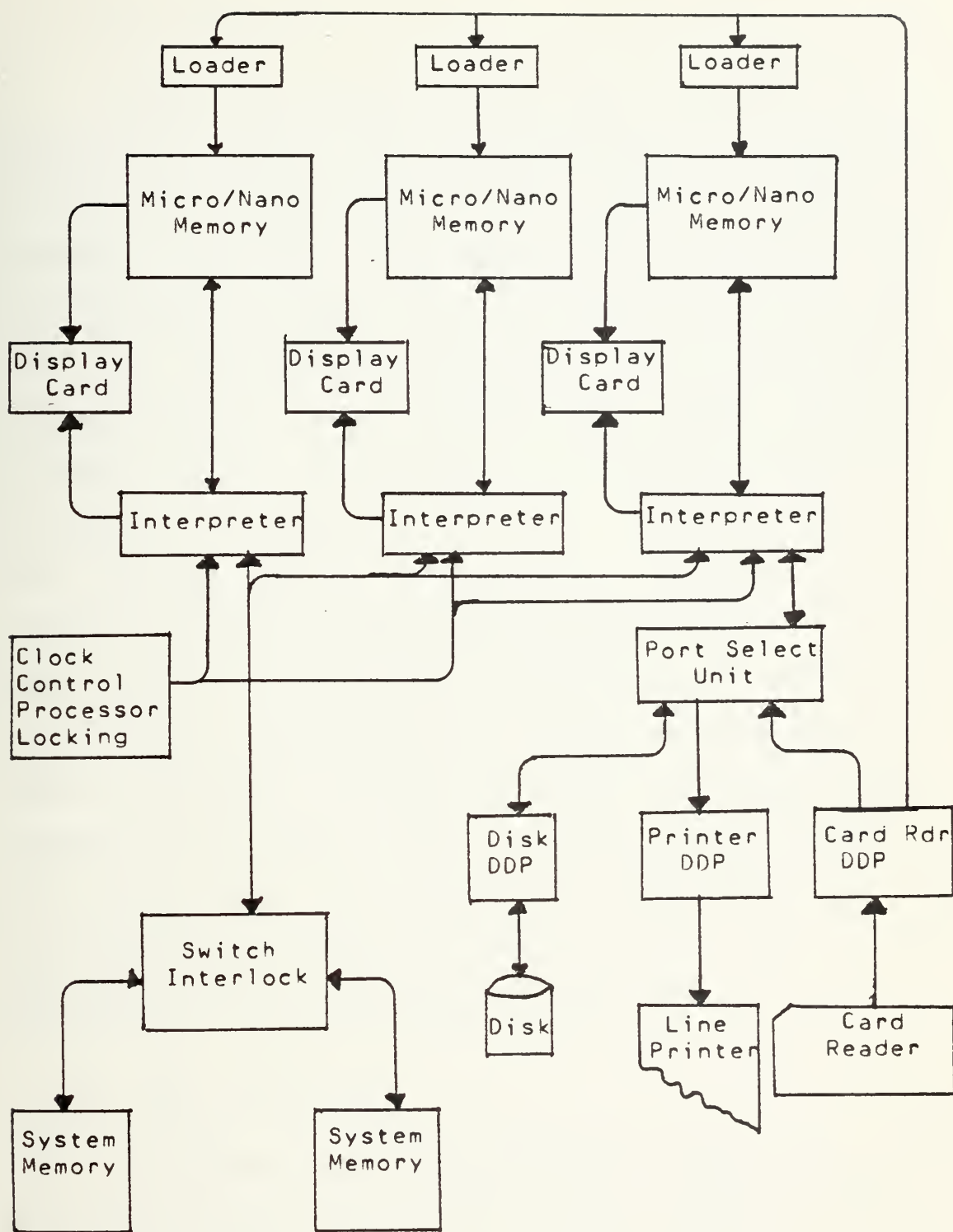
In general, there are three data processing functions to which the Interpreter may be applied: emulation of existing or hypothetical machines; direct execution of high level languages; and problem solution through microprogram "tuning" of the machine to the problem. The initial goal for the installation at the Naval Postgraduate School is emulation of existing machines such as the AN/UYK-7, with further expansion and development as a teaching tool in the areas of operating systems, file management, compiler writing, and emulation of hypothetical systems.

B. NAVAL POSTGRADUATE SCHOOL CONFIGURATION

1. Description

The system presently installed at the Naval Postgraduate School consists of three Interpreters, one 64K memory module, a card reader, line printer and dual cartridge disk. Only one of the three Interpreters is allowed to communicate directly with any peripheral and is discussed below under I/O Interface. Figure IV-2 represents the present configuration; however, future expansion calls for the addition of a supervisor's console and a cross-connection with the school's PDP-11/45. This will enhance the attached peripherals by three tape drives and greatly increase the amount of on-line storage to the point where implementation of a CMS-2 Compiler may be feasible.

When the system is energized and initialized daily, its basic microstructure is that of two B-6700 LIFO ALGOL Stack Machines - each with 32K of memory - attached to a single Input/Output Processor. The system is capable of pseudo-multiprocessing in this configuration. ALGOL programs submitted through the card reader are executed in a batch mode with the two processors competing to fetch the next job. All utility programs, the operating system and the file manager are executed with the machine in this configuration. In order to change the machine's configuration, the user must alter the microcode in order to execute the desired machine such as, the AN/UYK-7.



NAVAL POSTGRADUATE SCHOOL THREE INTERPRETER SYSTEM

FIGURE IV-2

2. I/O Interface

Since only one Interpreter (IOP) is allowed to exchange data directly with the peripherals, it is necessary for the remaining two processors to communicate their I/O requests to the IOP. This is done by the Interpreter placing a message in address 64K otherwise known as the "Mailbox", and then issuing an interrupt (INT) to the IOP. The IOP periodically tests for INT and when sensed, reads the Mailbox, decodes the message, and performs the predefined function. After completing its task, the IOP places a message in the Mailbox informing the requesting Interpreter whether or not the I/O was performed successfully. The IOP then issues an INT to the Interpreter which sent the request. When the requesting Interpreter receives the INT, it reads the Mailbox to determine whether its request was performed and continues accordingly. This method allows the two Interpreters to act completely independently, each referencing only its assigned segment of memory and the IOP performing all I/O asynchronously upon request.

3. Memory Interface

The memory module consists of a single ported, 64K, 32-bit word, core memory, which is the equivalent of 256K of on-line, 8-bit character storage. Access to this single memory is through a Switch Interlock Unit (SWI) which makes memory appear to be multiported to the user. The SWI acts on a priority basis with that Interpreter having the lowest

number (the IOP) given the highest priority. Once an Interpreter issues a memory read/write reference, it may continue execution and need not wait on a memory completion signal. However, the memory address register (MAR) on a Read/Write, and the Memory Information Register (MIR) on a write, should not be changed until a completion signal is received. Thus a microprogrammer who anticipates his memory accesses and intersperses these instructions among the other code, should never have a delay caused by memory referencing.

C. INSTRUCTION TIMING

The Interpreter uses a one megahertz clock and initiates a new microinstruction every microsecond. The Interpreter enhanced with Emitter Coupled Logic (ECL) and a higher speed memory can operate off an 8 Megahertz clock. A corresponding 8-fold improvement in execution times can be expected.

There are two basic instruction types in the Interpreter. Type I uses two phases (I and III) for execution; although, its phase III may be held in abeyance until completion of a subsequent Type II, which always uses only phase I to complete. Phase I of the following Type I instruction always overlaps Phase III of the previous Type I. Type I instructions involve condition testing, external functions and Adder/Logic operations. Type II instructions involve literal assignments to one of three registers (LIT, SAR and AMPCR). Appendix D of reference 9 contains an excellent discussion of instruction timing.

D. LANGUAGES

The D-Machine has two resident programming languages: the language of the operating system, ALGOL; and the microprogram assembly language, TRANSLANG.

1. ALGOL

The resident programming language of the Burrough's D-Machine is the ALGOL 60 Language enhanced with additional language constructs which permit manipulation of data in the form of character strings. ALGOL employs a vocabulary of reserved words and symbols. The structure of the language, syntax, reserved words and symbols, and additional features of the ALGOL implemented on the D-Machine are discussed in reference 10. The accepted character set for ALGOL varies depending on the machine used and the character set or sets available on the machine. The Naval Postgraduate School configuration requires that all characters be converted from EBCDIC into the 6-bit Burrough's Common Language (BCL) format for recognition by ALGOL. This conversion is performed by the IOP when it is used for interfacing communications from external devices to the other Interpreters. When information is directed to an external device the IOP performs a reverse conversion, from BCL to ASCII. Thus all inputs to the emulator from peripherals are 8-bit characters containing a 6-bit BCL code. The Interpreter's file management routines, operating system and utilities are written in ALGOL, for execution on the ALGOL Stack Machine

configuration. ALGOL source modules may be inputted to the ALGOL Compiler from disk or cards. Object modules are placed in user libraries on disk for future execution by the stack machine. The operating system (MCP) will load specific object modules for execution when it recognizes a "?RUN filename" control card, where filename is assumed to be a precompiled ALGOL program.

2. TRANSLANG

The microprogrammer is aided in producing microprograms by a Microtranslator/Assembler that translates symbolic instructions written in TRANSLANG into microinstructions. Reference 9 describes the structure of TRANSLANG by defining its syntax and semantics, and presenting a series of examples. The reference also includes descriptions of register state changes resulting from executing microinstructions. Interpreter controls and timing are explained. Coding techniques and conventions are discussed via sample programs. The reference analyzes external operations with main memory and peripheral devices, and the coordination and control of multiple Interpreters via the Switch Interlock and global condition bits.

The Microtranslator is written in ALGOL for the D-Machine. It is written modularly with each function setup as a procedure call. The language, TRANSLANG, used ALGOL as a model; however, almost the entire language is composed of reserved words. Reserved words have very specific meaning

to the translator and cause specific nanoinstructions to be developed. TRANSLANG is free form and each instruction may be written in almost any order; however, multiple instructions appearing on the same line or card must be separated by a period. Each TRANSLANG instruction corresponds to one microinstruction, which is the set of Interpreter functions performed in parallel at each machine clock. The constructs provided include iterative mechanisms, I/O, Boolean, logical and computational operations, control transfers, and assignment functions. In order to provide control points for transfer operations, each instruction may be labeled with a symbolic M-Address. The output module of the Microtranslator is placed on disk and when loaded into micromemory, alters the basic machine configuration. Reference 9 is used as the microprogrammer's programming manual; although, there have been several enhancements to the machine and language which are not included in the manual. These modifications are discussed in Appendix A of this thesis.

V. PROJECT DESCRIPTION

The AN/UYK-7 achieves its speed and versatility partially through concurrent field utilization and partially through the provision of special and general purpose registers. Its emulation is complicated by the various modes of instruction operation (repeat, indirection, indexing) and in particular by the use of ad hoc addressing techniques. The fact that the AN/UYK-7 performs many of its operations in parallel makes it difficult for an emulation which is imitating one facility at a time to run in AN/UYK-7 real-time. For the remainder of this discussion the acronyms A(a), B(b) and S(s) refer to the Accumulator, Index and Base registers, respectively. The subscripts (a, b, and s) correspond to the register specified in the cognizant field of the instruction, and may assume values of 0 to 7. Capital "Y" refers to the effective 18-bit, operand address; small "y" refers to the 13-bit, y-field of the instruction. The combination sy-field is formed by concatenation of the 3-bit, s-field and 13-bit, y-field of the instruction and forms an alternate operand address.

During the initial analysis of the Emulation it was decided to separate the project into the following phases:

First, the machine would be designed to accommodate a full emulation including multiprogramming and all I/O

functions. This required that all registers and condition bits of the actual AN/UYK-7 hardware be mapped into the model, even if they would not be used during this partial Emulation.

Second, it was decided to divide the Instruction Repertoire into the following groups:

- a. Those instructions that could be emulated immediately and tested completely; such as, all Format I's and II's except Multiply, Divide, Square Root and Double Register Operations.
- b. Those instructions that could be done prior to completion but for which there might be insufficient time for complete testing; such as, all Format III's, Shifts, Multiply, Divide, and Double Register Operations.
- c. Those special instructions and functional modes that would be incorporated if at all possible; such as, indirection and repeat.
- d. Those features for which there would definitely be insufficient time for incorporation; such as, Interrupts, IOC Instructions, and Floating Point Operations.

Third, a "Loader" program which could read AN/UYK-7 instructions from cards, decode them and place the results into memory for execution by the Emulator was mandatory.

Fourth, some facility for monitoring and debugging each instruction as it is executed by the Emulator would be required.

Fifth, some facility to output the results of each AN/UYK-7 program and to input test data was needed.

To accomplish these five tasks, two basic programs called the "Loader" and the "Emulator" were written. Each of these programs occupied its own Interpreter whenever the system was reconfigured as an AN/UYK-7.

A. LOADER

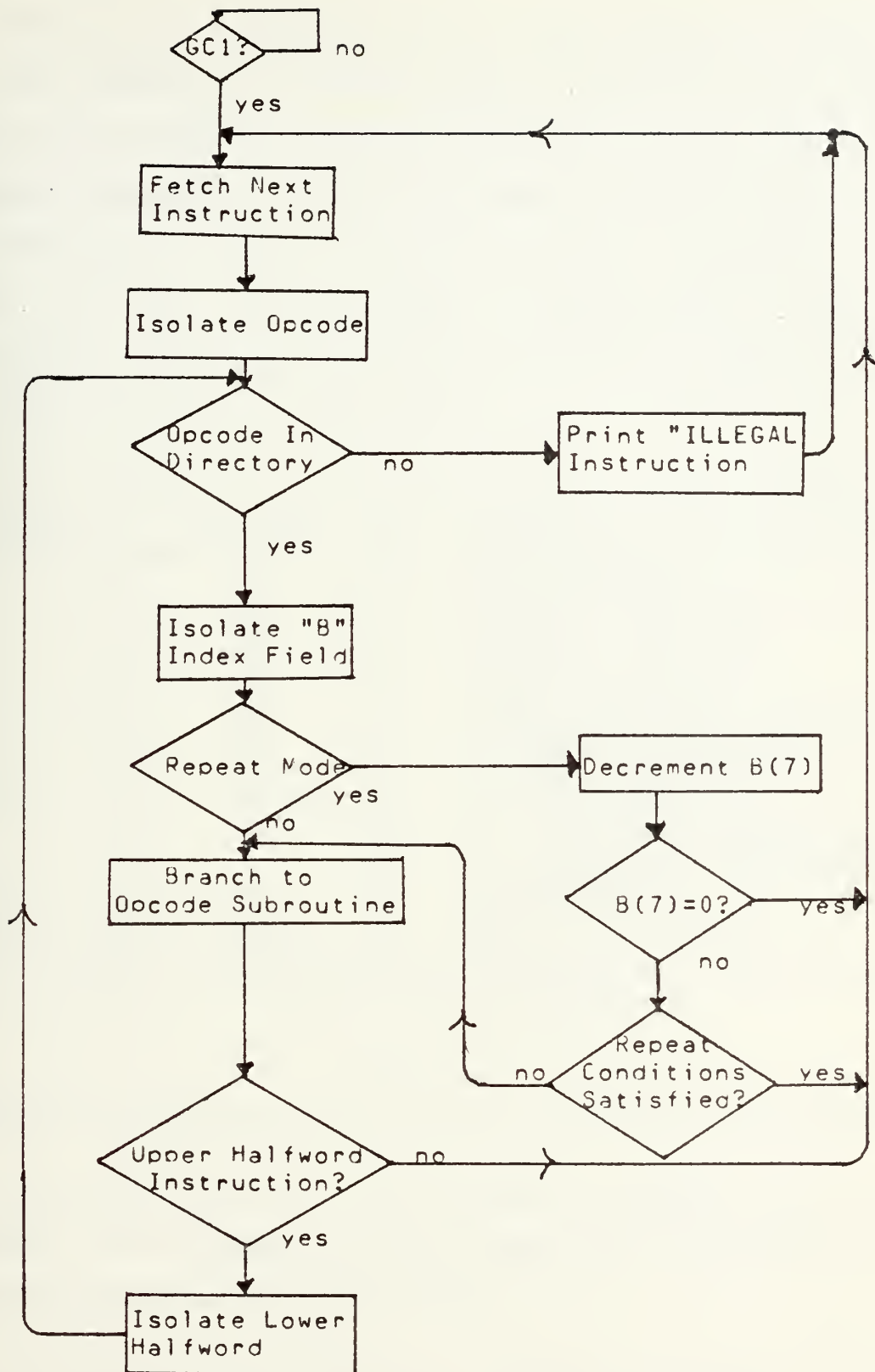
The Loader was written in TRANSLANG and fulfilled the third, fourth and fifth requirements; that is, it served as a combination loader, debugger and IOC monitor. Its use is described in Appendix A, the User's Manual section of this thesis. In general, the loader portion of the program acts like a pseudo-assembler. It has macros for defining a constant or character string, for saving space, for initializing registers and switches, and for inputting instructions. The loader will accept one instruction per card and determine whether it is a Format (I, II, III, IV-A, or IV-B) instruction. Based on this determination, the Loader will decode the specified fields from the card and generate a binary instruction which is placed in the next sequential location in the user's memory. Upon detecting an "N" card, the loader portion of the program signals the Emulator - which is resident in the alternate Interpreter - that instructions are available for execution.

At this point the loader function ceases and the program can be called upon by the Emulator to act as a debugger or as an IOC. As a debugger, the Loader can be used to generate a number of error messages to the operator, or to provide an actual dump of control memory registers 000 to 035, which includes all Task Registers, the PAR and the Active Status Bits. As an IOC, the Loader can be called upon to read a card, print a buffer, or read the disk.

It should be emphasized that the Loader was written strictly as a tool to assist in the development and testing of the Emulator. Presumably, once a full emulation is written and tested, the IOC functions will be incorporated into the Emulator, while the loader and debugger functions will be either written in AN/UYK-7 Machine Language, or absorbed by the Emulator. When this is done, the Interpreter previously utilized by the Loader will be free for loading as a second AN/UYK-7, and a true multiprocessing environment could be created. A copy of the Loader is provided in Appendix C.

B. EMULATION PROGRAM

The Emulator was written in TRANSLANG microcode, and is loaded directly into the Interpreter's micromemory from disk. A copy of the Emulator is provided in Appendix B. Overall program flow for the Emulator is presented in Figure V-1.



OVERALL PROGRAM FLOW

FIGURE V-1

Besides the necessary Control Memory Register mappings, it was necessary to minimize memory references by maintaining certain frequently used information in the D-Machine's internal registers. For this reason, the PAR is maintained in the lower 20 bits of the D-Machine's A2 register. The upper 12 bits of A2 are used to maintain some of the most frequently referenced bits of the AN/UYK-7's Active Status Register as listed in Table V-1.

| ASR Bit | A2 Bit | Function |
|---------|--------|--|
| 2 | 31 | Equal/Unequal |
| 1 | 30 | Greater Than/Equal |
| 0 | 29 | Limits |
| 3 | 28 | Fixed PT Overflow |
| 8 | 27 | Task use of 05-4 |
| 9 | 26 | Remove Interrupt Lockouts |
| 10 | 25 | Int/Task Accumulators |
| 11 | 24 | Int/Task Base Regs. |
| N/A | 22-23 | 01 = Indirection 10 = Optional Indirection 11 = Character Addressing |
| N/A | 21 | Repeat Mode |
| 15 | 20 | Halfword Indicator |
| N/A | 0-19 | PAR |

ACTIVE STATUS REGISTER

TABLE V-1

The remaining bits in the ASR are not used in this Emulation, but if needed could be mapped into any unused location in memory below address 1024.

The D-Machine's A1 register is used to maintain a copy of the instruction currently being executed. All other registers are free for computational use. The D-Machine's

General Condition Bit One (GC1) is reserved for communicating with the Loader. General Condition Bit Two (GC2) is used to indicate that an AN/UYK-7 instruction is being executed under a special condition; such as, repeat mode or indirection.

The Emulator consists of an Opcode/Sub-opcode Directory, an instruction fetch routine (IFETCH), global subroutines and opcode execution subroutines. When the Emulator receives a signal from the Loader (GC1) to commence execution, it immediately passes control to IFETCH, which reads the PAR and places the first instruction in A1. The opcode portion of the instruction is isolated and used as a pointer into the Opcode Directory. Control is then passed to the opcode execution subroutine designed to handle that particular instruction. The opcode subroutines may stand alone and perform their functions independently; or they may call several global subroutines used to consolidate code for multiple opcodes. Upon completion of its execution, the opcode subroutine passes control back to IFETCH which then fetches the next instruction.

In the case of halfword instructions, control is passed to HALFFETCH which serves the same purpose as IFETCH, except for 16 bit instructions. Every attempt is made within the opcode subroutines to do in-line coding versus subroutine calls. This decreases execution time considerably; although, it greatly increases program size. During the initial programming phase the Interpreters were limited to a

4K micromemory due to the dynamic overlay routine (SMX) being positioned at 4k. As a result, excessive subroutine calls were necessary in order to keep the program size below 4K. This software deficiency has since been corrected by Burrough's and a full 8K micromemory is now available. These subroutine calls could now be removed and the subroutines moved back in-line.

Emulation of Central Processor instructions in the AN/UYK-7 repertoire was fairly straightforward. Most instructions were implemented with an average of 15 microinstructions. This generally included: isolating the various fields, forming the effective operand (Y) address, fetching the operands from main memory, performing the instruction function, and writing the result back to main memory, commonly referred to as S-Memory. Because of the parallelism of the D-Machine's nanoinstructions several of these functions were performed simultaneously. Many AN/UYK-7 instructions have sub-opcodes, each of which perform a different operation on the same operands. In such cases, the opcode subroutine centralized code by performing the field isolation and operand fetch functions prior to calling the sub-opcode subroutine. Either the operand itself or its address was passed in a register to the subroutine, which then performed its function and wrote the results back to S-Memory.

The difficulties arose in the AN/UYK-7 Emulation when the repeat and indirection modes of operation were implemented. The Repeat Instruction required that the next

sequential instruction be executed the number of times contained in the Index Register B(7), or until specified alternate conditions were met. This required that flags be set indicating that the repeat mode had been initiated and that special termination conditions were to be tested. Condition testing took place after each execution of the repeated instruction and prior to return to IFETCH for the next instruction. If the conditions were not satisfied IFETCH was bypassed and the same instruction was reexecuted after first decrementing B(7), and then indexing the operand address by the displacement value contained in the repeat instruction. If the conditions were met the normal IFETCH mode was executed and the repeat mode was terminated. Most instructions required checking the condition of a specified accumulator register to signal completion of the repeat mode; however, compare instructions used the results of their comparison to signal completion. Repeated replace instructions used a different Y-operand calculation for their store phase then for their fetch phase. Instructions that were not designated as being repeatable were executed once and the repeat mode was terminated. A copy of the Repeat Instruction was saved in S-Memory at address 0030; a copy of the instruction being repeated was saved at address 0031; and the Y-operand address used for storing during a repeated replace instruction was saved at 0033. These addresses are given in octal and are normally unused in the AN/UYK-7.

The indirect modes of operation possible in the AN/UYK-7 were extremely difficult to emulate. There are four modes of indirection possible: Normal Indirection, Character Addressing, Sequential Character Addressing, and Optional Indirection. Not all instructions are indirectable and some instructions are indirectable but not character addressable. To make this determination a table was created in the Emulator which would return a value based on the instruction opcode. This value indicates whether an instruction is repeatable, indirectable, or character addressable. All modes of indirection are initially signalled by the i-field of the instruction being a one. This indicates that the Y-operand points to an Indirect Control Word (ICW). The ICW, which is presented in Figure III-1, is then fetched and its bits 30-31 are examined to determine the indirection mode. In all cases, if the i-field of the ICW is set, the Y-operand again points to a new ICW. The Y-operand is calculated in accordance with the indirect mode. Operand fetching will continue in this manner, cascading through memory, until an ICW is found without its i-field set. At this point the lower 20 bits of the ICW replaces the lower 20 bits of the original instruction.

Normal indirection is indicated by a binary 10 in bits 31-30 of the ICW. Y-operand address calculation is performed as $Y = y + B(b) + S(s)$. Once the final ICW is determined then normal execution of the instruction resumes.

Single Character Addressing is indicated by a binary 01 in bits 31-30 of the ICW. Y-operand address calculation is performed as $Y=y+B(b)+S(s)$; however, in this mode two additional fields (P,W) of the ICW are meaningful. The P-field indicates the least significant bit position of the character to be fetched from the operand and the W-field indicates the number of bits in the character. When the character is fetched - it is right justified; operated on in accordance with the instruction; and then (depending on the instruction) ORed back into its original position in the Y-operand.

Sequential Character Addressing indicated by a binary 11 in bits 31-30 of the ICW, operates in exactly the same manner as Single Character Addressing except in the case where the store cycle is required. Before ORing the resulting character back into the Y-operand the P and W-fields are compared. If P minus W is positive then the difference replaces P and the character is stored accordingly. If the difference is negative then 32 minus W replaces P and the "sy" field of the ICW is incremented by 1 and replaced in memory for the next execution.

Optional Indirection is indicated by a binary 00 in bits 31-30 of the ICW. In this mode bit 29 of the ICW is also significant and modifies the Y-operand address calculation. If bit 29 is 0 then $Y=sy+S(b)$. If bit 29 is 1 then $Y=sy+B(b)+(S)$, where S is designated by bits 17-19 of B(b). Once the first operand is fetched then normal instruction execution occurs.

A large portion of the Emulation code was devoted to handling the four modes of indirection and its ad hoc address calculations. General Condition Bit 2 (GC2) was used to flag the indirect and repeat modes of operation. Address 0032, octal, in S-Memory was reserved for saving the current ICW being used during Character Addressing; address 0033, octal, was reserved for the Y-operand address in the case of optional indirection; address 0034, octal, was reserved for the address of the current ICW which could be updated during sequential character addressing. These addresses are part of the Control Memory Registers and are not normally used in the AN/UYK-7.

Although the IOC has not been emulated, Input/Output functions were implemented which make the systems' peripherals accessible to the user. The AN/UYK-7 Initiate I/O instruction was implemented in the Emulator with the exception that the a-field, which is normally used as a channel designator, actually designates the desired peripheral. Based on the contents of the a-field, a function code is formulated and passed to the IOC portion of the Loader. The Emulator then enters a neutral state awaiting an Interrupt from the Loader that the I/O has been performed. The Y-operand address of the I/O instruction fetched by the Emulator points to the address of the buffer, rather than to a buffer control word as in the AN/UYK-7. It was felt that

implementation of I/O in this manner would remain virtually transparent to the user, and would facilitate full emulation of IOC functions at a later date.

Of the 132 instructions in the AN/UYK-7 Repertoire, 111 have been fully implemented and tested. One instruction (Return Jump) has been implemented but not tested. Twenty instructions remain to be both written and tested. All unwritten instructions were assigned space in the Opcode Directory and, if used by the programmer, will cause an error message to be printed to the effect that the instruction has not been yet implemented. The implementation of these instructions involves developing the required microcode and inserting it into the space already allocated in the Opcode Subroutine Library. The unimplemented instructions include all Floating Point Operators, Scale Factor, and Interrupt Handlers.

The Repeat Instruction and its associated mode of operation have been implemented and tested with the exception of the repeat replace instructions, which have been written but not yet tested. Indirect addressing has been fully implemented but has not been tested. Input/Output functions of the IOC have been implemented and tested allowing the user to read a card, print a buffer, or read a disk sector. The disk write function has been designed into the Emulator but not implemented for fear of destroying resident disk material during execution. This function should not be fully employed until either a monitor system is developed which

would preclude write access to assigned disk sectors, or the assumption is made that the entire disk is available to the user as a scratch pad.

C. REGISTER MAPPING

Because the D-Machine has few registers available to the user, the authors chose to map all of the AN/UYK-7 registers into main memory. In addition, it was necessary to create several special locations to hold temporary results or conditions. In order to facilitate this mapping, all memory locations below 1024 decimal are reserved; thus, all user instructions and memory references were offset by this amount by the Loader. Since this Emulation will be in a monoprogramming environment, all Base Registers were initialized in increments of 8K with the first Base Register set to 1024. This allowed user access to approximately 63K of main memory, with each Base Register referencing an 8K page of memory. The Control Memory Register (CMR) allocations and additional temporary mappings (indicated by an *), are presented as part of Figure V-2. The registers marked "*" are not used in the AN/UYK-7, and were thus assigned special purposes in the Emulator. Additional unused register areas are available and labelled "unused" in Figure V-2. These register areas could be employed for the same purposes if needed in the course of expansion to a full emulation.

| OCTAL ADDRESS | DECIMAL ADDRESS | DESCRIPTION |
|---------------|-----------------|---|
| 0-7 | 0-7 | Task A-Reg. (0-7) |
| 10 | 8 | Unused |
| 11-17 | 9-15 | Task Index B-Reg. (1-7) |
| 20-27 | 16-23 | Task Base S-Reg. (0-7) |
| *30 | 24 | Repeat Instruction Temp. |
| *31 | 25 | Current Instruction being Repeated |
| *32 | 26 | Current Indirect Control Word (ICW) |
| *33 | 27 | Y-operand for Indirection |
| *34 | 28 | ICW Address in Memory |
| *35 | 29 | Program Address Register |
| 36-57 | 30-47 | Unused |
| 60-67 | 48-55 | Interrupt Breakpoint Registers (0-7) |
| 70-77 | 56-63 | Active Status Words (0-7) |
| 100-107 | 64-71 | Int. A-Registers (0-7) |
| 110 | 72 | CP Monitor Clock |
| 111-117 | 73-79 | Int. Index B-Reg. (1-7) |
| 120-127 | 80-87 | Int. Base S-Registers (0-7) |
| *130 | 88 | Switch Values Temporary |
| 131-137 | 89-95 | Unused |
| 140-157 | 96-103 | Designator Storage Words (DSW) |
| 160-167 | 104-111 | Storage Protection Registers (SPR) |
| 170-177 | 112-119 | Segment Identification Registers (SIR) |
| *200 | 120 | Location Counter |
| *201 | 121 | Real Time Clock |
| *202-246 | 122-166 | Disk Input Buffer |
| *247-323 | 167-211 | Disk Output Buffer |
| *324-357 | 212-231 | Card Input Buffer |
| 360-365 | 232-237 | Unused |
| *366-426 | 238-270 | Printer I/O Buffer |
| 427-437 | 271-279 | Unused |
| *440-1777 | 280-1023 | Error Routines |
| 2000-77775 | 1024-65533 | User Program Area |
| *77775-77776 | 65534-65535 | Mailbox |

*-Temporary Location Established By Programmers Convention

REGISTER - MEMORY MAPPING

Figure V-2

D. TIMING

Although no formal timing survey was conducted, an analysis of the Emulation itself provided an indicator which was used in developing a comparison to real-time. For this analysis it was assumed that each instruction was executed using direct addressing and not in repeat mode. The IFETCH routine used 16 microseconds which was an automatic overhead acquired by all instructions. The operand fetch and store routines (YFETCH, YSTORE and Y2FETCH) added an additional 33, 30, or 14 microseconds, respectively. Thus any instruction which referenced a Y-operand incurred an additional overhead of 14 microseconds at a minimum. Adding to this overhead, the execution time of the basic opcode itself yielded the time estimated for typical instructions given in Table V-2.

| INSTRUCTION TIME | OPERANDS | EMULATION TIME | AN/UYK-7 TIME |
|---------------------|------------|-------------------|------------------|
| Load | Register,Y | 60 | 1.5 |
| Add | Register,Y | 63 | 1.5 |
| Add | Register | 36 | 1.0 |
| Divide | Register,Y | 600 | 15 |

TIME ESTIMATES

TABLE V-2

Using these figures it was estimated that this Emulation ran on the order of 40 times real-time. Two steps could be taken that would considerably improve this time. First, move all out-of-line subroutine calls in-line in the opcode subroutines. Second, enhance the Interpreter as discussed under Instruction Timing in Chapter III. It has been estimated that this same Emulation could run at approximately 5 times real-time on an enhanced D-Machine. This time compares favorably with a Simulation of the AN/UYK-7 written in PL-360 for the IBM 360, which runs on the order of 1000 times real-time. As discussed in Chapter II, it would be difficult to approach real-time in emulating the AN/UYK-7 because of the parallelism of its fetch and field isolation operations. The addition of a Field Select Unit (FSU) and additional internal temporary registers would be required before a real-time emulation could be realized.

VI. SUMMARY

The authors' conclusions and recommendations resulting from any research project are a fundamental part of the formal presentation and are therefore included in this thesis. In addition, the authors felt that the reader, who might be contemplating an emulation project, would be interested in the problem areas encountered during the pursuit of this thesis. Accordingly this chapter is divided into two sections - Problems and Conclusions.

A. PROBLEMS

During the course of this emulation numerous software and hardware problems were encountered which had a considerable effect on progress and the amount accomplished in the allocated time. Software problems stemmed principally from poor documentation of the D-Machine, the AN/UYK-7, their associated languages and internal configuration. The software problems with the D-Machine were generally resolved through experimentation on the machine or through phone calls to Burrough's Advanced Development Organization (ADO). The lack of thorough AN/UYK-7 documentation proved to be a more severe handicap since no test base for experimentation was accessible at the Naval Postgraduate School. Such a medium is absolutely mandatory when working in the emulation

environment. The programmer must be able to determine the machine reaction to unorthodox instruction configurations. Some degree of assistance was provided by the Fleet Combat Director Systems Support Activity (FCDSSA), San Diego; however, in some cases the result was a "best guess" as to what the AN/UYK-7 response would be.

Hardware problems are to be expected during the course of any major project and particularly when a new machine is involved. This Emulation was the first project to be undertaken on the Naval Postgraduate School's D-Machine since the machine was installed. The estimated mean-time-between-failures (MTBF) experienced by the authors was less than five hours. Problems ranged from loose circuit boards and dirty contacts, to disk read/write head misalignment. Unfortunately, the time to repair was aggravated by the fact that the school has no technicians trained on the D-Machine, and the authors were inexperienced in its maintenance and operating procedures. Without the assistance provided by Burrough's ADO the project could not have been seen through to fruition. The ADO provided the school with engineering assistance on three separate occasions, the last of which uncovered a major design problem. The machine has since been modified and the system is presently very reliable.

In addition to these two major problem areas progress was inhibited by the following:

1. Lack of Software Utilities - Since the D-Machine was a new installation the school has not had the opportunity to generate the numerous desirable software utilities. In an effort to eliminate the inherent slowness of a "Card Only" system, the authors have undertaken the implementation of the supervisor's console as an adjunct to this thesis. The authors are also implementing a Text Editor which will allow for on-line program modification, and will replace the present card input Line Editor.
2. Microprogramming is not a new concept at the Naval Postgraduate School; however, actual experience was lacking. The Computer Science Department has played a major role in implementation of the D-Machine and the required expertise is rapidly being developed.
3. The learning curve for undertaking an emulation is tremendous. The target machine, the host machine, their respective programming languages and internal configurations must be learned and understood in minute detail.
4. The target machine for this Emulation, the AN/UYK-7, is an extremely sophisticated and difficult to understand computer. The central processor instruction set is fairly straightforward and easy to implement; however, the numerous and differing modes of operation severely complicated the Emulation.

B. CONCLUSIONS

The authors feel that they have successfully achieved their goal of demonstrating the feasibility of emulating the AN/UYK-7 on the Burrough's D-Machine. The design presented in this thesis is a workable design as evidenced by execution of AN/UYK-7 programs successfully and by execution in

less than 40 times the actual execution time of the AN/UYK-7. Better than 90% of the AN/UYK-7 Instruction Repertoire has been implemented and tested, thus providing a sound foundation for a full emulation. The design allows for expansion to a full emulation which would include the IOC instruction repertoire and Floating Point without major modification. Further, the authors have concluded that by modification of the design to place out-of-line subroutine calls in-line and enhancement of the D-Machine, the Emulation would execute within 5 times the actual speed of the AN/UYK-7.

During the course of this project the authors had an opportunity to visit the Naval Surface Weapons Center (NSWC), Dahlgren, Virginia. Emulation in general was discussed with the programming staff, who had just completed emulation of the Trident Computer on the Nanodata QM-1, and who were contemplating emulating the AN/UYK-7 on the QM-2. Based on these discussions, the experience gained in the course of this thesis, and presupposing the present level of knowledge of the authors; it is estimated that a complete emulation of the AN/UYK-7, its IOC and all Interrupt functions would require an additional 2 man years of programming effort. This estimate includes a reasonably sophisticated degree of testing, and assumes full accessibility of the D-Machine and an AN/UYK-7 test base.

VII. RECOMMENDATIONS

The authors feel that this thesis has provided them with a learning experience previously unparalleled in either of their lives, since it has entailed detailed and independent learning of emulation, microprogramming (TRANSLANG), the Burrough's D-Machine and the AN/UYK-7. In addition, the authors became part time technicians in an effort to keep the D-Machine running. It is felt that the same educational horizons await any student considering a similar project; however, in an effort to keep others from having to "reinvent the wheel", the authors have included much of what was learned about these subjects in this thesis. Hopefully, by building on these experiences, others will be able to begin at a much higher level and consequently accomplish more. The following recommendations are submitted for consideration by anyone inclined to undertake a thesis involving emulation.

1. Do not try to emulate a machine that uses hardware interrupts on an Interpreter that does not have interrupts.
2. Pick a target machine that is well defined and documented. Computers are designed to perform a specific set of functions; the results of unorthodox use are not always known, but must be considered by the emulator. The availability of a target machine for testing to determine such results is mandatory.

3. Ensure that the host Interpreter is well documented and has a reasonable set of system utilities; such as, an editor, debugger, simulator, file manager, operating system and compiler.
4. Ensure the host Interpreter will receive hardware support if problems arise.
5. Ensure publications are available covering utilities, hardware, languages and target/host machine capabilities and operation.
6. Be prepared to work independently and have your questions answered through your own experimentation.

In addition to these recommendations, the authors would like to propose several thesis topics applicable to the D-Machine.

1. Several projects could be undertaken as a direct follow on to this thesis:
 - a. An Operating System, File Manager, and Assembler could be written in AN/UYK-7 machine language utilizing the existing Emulator.
 - b. The present Emulation could be enhanced to include Floating Point and any unimplemented instructions, including incorporation of the IOC functions and allowing for synchronous I/O.
 - c. A timing survey could be conducted comparing the existing Emulation with AN/UYK-7 benchmarks.
2. Several thesis projects are possible which entail enhancing the capabilities of the existing D-Machine Installation. These include writing the interface to the PDP-11/45, and a resident Text Editor.

The stand alone environment of the D-Machine lends itself to experimentation with the design of operating systems and file managers.

In summary, the authors would like to state that they found the D-Machine an outstanding learning device in spite of its many hardware problems. The authors found the ability to emulate existing computers exciting, and found the realization that they could build a working model of any computer they could design, exhilarating.

APPENDIX A. USER'S MANUAL

A. D-MACHINE

When the D-Machine was installed at the Naval Postgraduate School, it was not accompanied by any form of documentation for its operation, preventive maintenance, diagnostics or utility programs. For the most part this information has been derived through experimentation and frequent calls to Burrough's ADO. Some of the more pertinent information is included in this User's Manual to enable the user to load and run a program such as the AN/UYK-7 Emulation. Unless specified otherwise all addresses given in this discussion are in hexadecimal.

1. System Initializing

The machine is secured every night and must therefore, be reinitialized when powered up each morning.

- a. Ensure circuit breakers 2, 7 and 10 are on.
- b. Push the "ON" button on the disc and all three Interpreters.
- c. With desired discs in the drives, push the two "RUN" buttons on the disk unit.
- d. Turn on the card reader and printer.
- e. Place the IOP in normal vice single step mode via the switch inside the cabinet on the right.

- f. Push the "LOAD" button on the IOP; then push "CLEAR".
- g. Place the cards marked IOP-Loader into the card reader. Cards should be read and the IOP should stop at an MPAD of 00FA. If not, repeat steps e through g.
- h. Push the "LOAD" button on IOP to return the IOP to normal.
- i. Perform steps e to h for each of the other two Interpreters using the cards marked STK-loader. They should halt at a MPAD of 004A.
- k. Push "CLEAR" on each Interpreter. They should bootstrap themselves through the IOP and halt at either 0542 or 051A. (An Interpreter must be at 0542 to run a program and only one Interpreter can be at 0542 at a time.) Clear the IOP to force an Interpreter to switch from 051A to 0542.

The D-Machine now thinks it is a Burrough's B6700 single-stack machine. Any utility can be run using the proper control cards as documented in the computer lab. The more common utilities are Sunrise (updates the day, date and time; should be run first thing every morning), Dir-List (prints out the directory; system or user), Compile (compiles ALGOL programs) and TRANSLANG (assembles microprograms). By use of the proper control cards, the user can also obtain a source listing of his program as it is being compiled or assembled. Actually the default is a printout every time and the user must insert a "\$-list clist" control card to eliminate the listing.

To make the computer act as some machine other than the B6700, use the control card "?SMLoad filename". The "?" must be in column one and the rest of the card is free-format. This loads the microprogram "filename" (which must be an assembled TRANSLANG program) into micro-memory overlaying the stack machine and then transfers control to that program starting at address 0000. Thus to load the AN/UYK-7, execute "?SMLoad UYK7-LOADER" on one Interpreter and "?SMLoad ANUYK7-EMULATOR" on the other. Refer to the appropriate sections of this chapter for further instructions on the execution of these programs.

2. Diagnostics

The programs used for diagnosing hardware problems are written in TRANSLANG and maintained as object modules on cards. The appropriate program is loaded in the same fashion as the bootstrap loader and terminates at a significant address. When "cleared", it should again terminate at an address which indicates success or failure. The operator must determine which from the program listing, as no messages are printed. There are no diagnostics written for the disk, printer or card reader. At the present time a CRT terminal is hooked up to the IOP; however, there is no software support for it. An interface is being written into the operating system and eventually all commands to and from the operator will be via the CRT. The card reader and line printer will then be strictly data I/O ports.

3. Debugging

While executing any TRANSLANG program, the user can monitor the program by use of the nixie light display panel and the function switch assigned to each processor. The lights indicate hex numbers. By selecting the appropriate function the user can observe the current nanoinstruction being executed; the address of the current microinstruction (MPAD); the memory address last written to or read from (BMAR); and the MIR register (MIR and EXT). The MIR function displays the lower 16 bits of the MIR register while the EXT function displays the upper 16 bits. These functions are presently the only means of debugging. For example, if the user were unsure of the information being read from S-memory, he could insert two additional instructions in the code: an instruction to place the data into the MIR register and a "WAIT" instruction. The CPU halts at the "WAIT" instruction and the user can then examine the information in MIR to ascertain its validity. The user may check the MPAD to determine if the address read was actually the address intended. To continue the program the user must press the force-step button inside the side panel. If the user wished to step through a series of instructions following the "WAIT", he could go to single-step, and while holding the force-step button, press the single-step button to override the "WAIT". After passing the "WAIT", single-step to execute one instruction at a time. In this manner the user can debug several instructions one at a time.

B. AN/UYK-7

This section of the User's Manual discusses how the user can reconfigure the D-Machine into an AN/UYK-7.

First, ensure that the disk packs which are mounted are the AN/UYK-7 System Packs and that both interpreters are loaded with an ALGOL Stack Machine.

Second, insert an "?SMLOAD UYK7-LOADER" card in the card reader. This will cause the Loader's microcode to overlay the stack machine's microcode and stop at address "0000".

Third, insert an "?SMLOAD UYK7-EMULATOR" card in the reader. This will cause the Emulator's microcode to overlay the alternate stack machine and stop at address "00A0". The AN/UYK-7 is now ready to accept and execute programs.

Fourth, place the AN/UYK-7 source programs in the reader and force step the Loader and the Emulator, respectively. The Loader will read one card at a time, decode it and place the 32-bit resulting instruction in memory for subsequent execution by the Emulator.

The last instruction of all user programs should be a "HALT", which will cause the Emulator to halt and the Loader to be reinitialized for the next job. Force stepping the Emulator after the halt will cause the next job to be read and executed. Use of this convention permits batch processing of programs by the AN/UYK-7. The program deck organization is discussed in the next section. Individual programs are separated by "N" cards.

1. Loader

a. Macros

This section describes how a program must be keypunched for input to the Loader. All Macro control characters and formats are listed in Figure A-1. Any character typed in column one, which is not listed, will cause an "Illegal Instruction" message and the card will be ignored. It is noteworthy that all functions are optional with the exception of the "N", which must terminate the program deck. All addresses are in octal and considered absolute as far as the user is concerned; however, they are offset by 1024 by the Loader. Macros and instructions may be interspersed as desired.

| COL. 1 | COL. 4-8 | FUNCTION |
|-------------------|-------------|---|
| "A" Register 0-7 | | Cols. 9-19 contain the octal value to be inserted in the listed A-Reg. |
| "I" Register 1-7 | | Cols. 9-19 contain the octal value to be inserted in the listed B-Reg. |
| "D" Octal Address | | Cols. 9-16 contain the decimal value to be inserted at the address listed or in-line if the address=0000. |
| "R" Octal Address | | Cols. 9-19 contain the decimal number of words to be saved at the address listed or in-line if the address=0000. |
| "W" Octal Address | | Cols. 9-80 contain the character string to be inserted at the address specified. The string terminates before col. 80 if a single quote (') is encountered. |
| "S" Switch Value | | Cols. 6-8 contain an octal number whose bit pattern selects which of eight AN/UYK-7 switches the user desires set. |
| "L" Octal Address | | Causes Base S-Registers to be set at 8K pages starting at 1024, and the program to be loaded at Address. |
| "O" Octal Address | | Causes change in the program counter in order that subsequent instructions may be loaded at Address. |
| "N" Octal Address | | Terminates loading of program and initializes PAR to Address for the Emulator. |

LOADER MACRO DEFINITIONS

FIGURE A-1

b. Instruction Preparation

For inputting instructions to the Loader three basic formats are used which correspond roughly to the five formats used by the AN/UYK-7. The Loader formats are presented in Figure A-2 under their respective AN/UYK-7 Format Headings. All fields require a right justified octal representation. The "i" field should always be either a one or a zero since it represents a single bit. The combined "F3,K" field of Format III instructions requires that K always be zero; therefore, valid inputs are (0,2,4,6) which correspond to an F3 of (0,1,2,3). Columns 1-3 must always be left blank. Columns 15-80 may be used for programmer's comments. Each instruction is decoded into one 32-bit word and assigned to the next sequential address in memory. However, if the instruction is a half-word instruction then it will be stored in the lower half of the previous word if that word also contained a half-word instruction, else it will be stored in the upper half of the next sequential address. An instruction's sequential address assignment can only be changed via the "O" or "L" macros, which must precede the instruction. A sample program is presented in Appendix D.

FORMAT I

| | | | | | | |
|-------|--------|-------|-----|-----|-----|-----------|
| COL | COL | COL | COL | COL | COL | COL |
| 1-3 | 4-5 | 6 | 7 | 8 | 9 | 10-14 |
| BLANK | OPCODE | A-REG | K | B | I | Y-OPERAND |

FORMAT II

| | | | | | | |
|-------|--------|-------|-----|-----|-----|-----------|
| COL | COL | COL | COL | COL | COL | COL |
| 1-3 | 4-5 | 6 | 7 | 8 | 9 | 10-14 |
| BLANK | OPCODE | A-REG | F2 | B | I | Y-OPERAND |

FORMAT III

| | | | | | | |
|-------|--------|-------|------|-----|-----|-----------|
| COL | COL | COL | COL | COL | COL | COL |
| 1-3 | 4-5 | 6 | 7 | 8 | 9 | 10-14 |
| BLANK | OPCODE | A-REG | F3,K | B | I | Y-OPERAND |

FORMAT IV-A

| | | | | | |
|-------|--------|-------|-----|-----|-----|
| COL | COL | COL | COL | COL | COL |
| 1-3 | 4-5 | 6 | 7 | 8 | 9 |
| BLANK | OPCODE | A-REG | F4 | B | i |

FORMAT IV-B

| | | | |
|-------|--------|-------|------------------|
| COL | COL | COL | COL |
| 1-3 | 4-5 | 6 | 7-9 |
| BLANK | OPCODE | A-REG | SHIFT DESIGNATOR |

INSTRUCTION FORMATTING

FIGURE A-2

c. TRANSLANG

Several modifications have been made to the D-Machine and the microprogramming language, TRANSLANG, since the publication of reference 9 in 1970. Unfortunately, an addendum to reference 9 has not been published and these powerful modifications remain undocumented. Because the authors took advantage of these capabilities in writing this Emulation the changes are documented here to provide the user a ready reference. They include: additional logical operators, program address modifiers, and an additional Y-select input register (BMAR).

(1) Logic Operators. The additional logic operators are listed in Table A-1 along with the code generated by the Microtranslator when they are encountered. In the Table, A1 and B are assumed as the X/Y input operands respectively. In addition, listed under the columns labelled TRUE/FALSE, are the tests generated by the Microtranslator when a TRUE/FALSE conditional test is encountered subsequent to the logical operation.

| OP | MEANING | CODE GENERATED | TRUE TEST | FALSE TEST |
|-----|-----------------------------|-------------------|--------------|---------------|
| LSS | Less Than | A1-B | NOT AOV | AOV |
| LEQ | Less Than or Equal | A1-B-1 | NOT AOV | AOV |
| EQL | Equal | A1 EQV B | ABT | NOT ABT |
| NEQ | Not Equal | A1 EQV B | NOT ABT | ABT |
| GEO | Greater Than or Equal To | A1-B | AOV | NOT AOV |
| GTR | Greater Than | A1-B-1 | AOV | NOT AOV |

LOGICAL OPERATORS

TABLE A-1

The programmer is cautioned that when the Microtranslator encounters the keywords TRUE/FALSE it backtracks to the last logical operator to determine which test to generate. Thus a TRUE/FALSE conditional test which is arrived at through branching may not have generated the exact code the programmer anticipated. These operators were intended to be used in a sequential fashion as follows:

```
A1 GTR B
If TRUE Then (operation)
```

The Microtranslator generates:

```
A1-B-1
If AOV Then (operation)
```

The following illustrates circumstances in which the Microtranslator does not generate the code expected by the programmer:

```
A1 LSS B
If LC1 Then A1 GTR B;Step Else Skip
If TRUE Then (operation)
```


Generates the code:

```
A1-B  
If LC1 Then A1-B-1;Step Else Skip  
If AOV Then (operation)
```

In this situation "If AOV" is always generated when the TRUE is encountered; although, obviously if LC1 was not set then the programmer desired to test for "Less Than".

(2) Program Address Modifiers. Two powerful new instructions were added to the D-Machine which allow direct transfer of program control. These are the two Type II instructions: "Label-1=MPCR" and "Label-1=CPCR". The first causes a direct change in the program sequence by loading MPCR with the new Label. The second can be classified as a return jump since it causes a transfer of AMPCR to MPCR, and MPCR+1 to AMPCR; thereby setting up a program jump to a subroutine, and saving the return address. This code is identical to:

```
Label-1=AMPCR  
CALL
```

The only difference between them is that the CPCR requires only one instruction for execution, while the CALL requires two instructions.

(3) BMAR. The BMAR register is formed by the concatenation of BR1 or BR2 (16 bits) and MAR (8 bits). It can be used as a Y-Input of 24 bits to the Adder. The actual BR register selected is the one most recently referenced

in an external operation. The user is cautioned in the use of BMAR because of its odd size of 24 bits. The unwary programmer could unintentionally allow extraneous bits in the most significant byte when BMAR is used for temporary storage of 16-bit addresses. That is, the assignment "A1=MAR" is considered to effect a transfer of 16 bits, while in fact 24 bits are transferred. A subsequent statement of "BMAR=A1" could permit extraneous bits to be introduced in the third least significant byte, if the programmer had made the assumption that only 16 bits were involved and automatic masking had taken place. The programmer can force selection of BR1 or BR2 by issuing an ASR or ASE, respectively, prior to a BMAR reference. These are considered external function NOOPS in the Naval Postgraduate School Configuration; however, they may not appear concurrently with a BEX statement.

The remainder of the TRANSLANG statements are reasonably straightforward once the programmer understands the instruction timing idiosyncracies. Reference 9 provides excellent guidance for the novice microprogrammer.

APPENDIX B. EMULATOR PROGRAM LISTING

This appendix provides a copy of the Microtranslator output listing of the Emulator. A copy of the source program is maintained on cards and also on the Burrough's D-Machine disk. The object module produced by the Microtranslator is maintained on disk. Each line of the program is divided into four sections. The leftmost grouping consists of the hexadecimal address to which the microinstruction was assigned by the Microtranslator during assembly. This is followed by the 56-bit microinstruction which was generated. The middle group consists of the programmers' source coding which was input to the Microtranslator. The rightmost group consists of the sequence numbers which were assigned to each input card as it was inserted into the source file on the disk by the Interpreter's resident line-editting program (CARD-LIST).


```

0061 0610 0000 0000 0000
0062 0620 0000 0000 0000
0063 0630 0000 0000 0000
0064 0640 0000 0000 0000
0065 0650 0000 0000 0000
0066 0660 0000 0000 0000
0067 0670 0000 0000 0000
0068 0680 0000 0000 0000

0069 0690 0000 0000 0000
006A 06A0 0000 0000 0000
006B 06B0 0000 0000 0000
006C 06C0 0000 0000 0000
006D 06D0 0000 0000 0000
006E 06E0 0000 0000 0000
006F 06F0 0000 0000 0000
0070 0700 0000 0000 0000

0071 0710 0000 0000 0000
0072 0720 0000 0000 0000
0073 0730 0000 0000 0000
0074 0740 0000 0000 0000
0075 0750 0000 0000 0000
0076 0760 0000 0000 0000
0077 0770 0000 0000 0000
0078 0780 0000 0000 0000

0079 0790 0000 0000 0000
007A 07A0 0000 0000 0000
007B 07B0 0000 0000 0000
007C 07C0 0000 0000 0000
007D 07D0 0000 0000 0000
007E 07E0 0000 0000 0000
007F 07F0 0000 0000 0000
0080 0800 0000 0000 0000

0081 0810 0000 0000 0000
0082 0820 0000 0000 0000
0083 0830 0000 0000 0000
0084 0840 0000 0000 0000
0085 0850 0000 0000 0000
0086 0860 0000 0000 0000
0087 0870 0000 0000 0000

0088 0880 0000 0000 0000
0089 0890 0000 0000 0000
008A 08A0 0000 0000 0000
008B 08B0 0000 0000 0000
008C 08C0 0000 0000 0000
008D 08D0 0000 0000 0000
008E 08E0 0000 0000 0000
008F 08F0 0000 0000 0000

0090 0900 0000 0000 0000
0091 0910 0000 0000 0000
0092 0920 0000 0000 0000
0093 0930 0000 0000 0000
0094 0940 0000 0000 0000
0095 0950 0000 0000 0000

OPR07XX:
OPR07X0-1=AMPCR
OPR07X1-1=AMPCR
OPR07X2-1=AMPCR
OPR07X3-1=AMPCR
OPR07X4-1=AMPCR
OPR07X5-1=AMPCR
OPR07X6-1=AMPCR
ILLEGAL-1=AMPCR

OPR50XX:
OPR50X0-1=AMPCR
OPR50X1-1=AMPCR
OPR50X2-1=AMPCR
OPR50X3-1=AMPCR
ILLEGAL-1=AMPCR
ILLEGAL-1=AMPCR
ILLEGAL-1=AMPCR
ILLEGAL-1=AMPCR

OPR51XX:
OPR51X0-1=AMPCR
OPR51X1-1=AMPCR
OPR51X2-1=AMPCR
OPR51X3-1=AMPCR
ILLEGAL-1=AMPCR
ILLEGAL-1=AMPCR
ILLEGAL-1=AMPCR
ILLEGAL-1=AMPCR

OPR52XX:
OPR52X0-1=AMPCR
OPR52X1-1=AMPCR
OPR52X2-1=AMPCR
OPR52X3-1=AMPCR
ILLEGAL-1=AMPCR
ILLEGAL-1=AMPCR
ILLEGAL-1=AMPCR
ILLEGAL-1=AMPCR

OPR53XX:
OPR53X0-1=AMPCR
OPR53X1-1=AMPCR
OPR53X2-1=AMPCR
OPR53X3-1=AMPCR
ILLEGAL-1=AMPCR
ILLEGAL-1=AMPCR
ILLEGAL-1=AMPCR
ILLEGAL-1=AMPCR

OPR70XX:
OPR70X0-1=AMPCR
OPR70X1-1=AMPCR
OPR70X2-1=AMPCR
OPR70X3-1=AMPCR
ILLEGAL4-1=AMPCR
ILLEGAL4-1=AMPCR
ILLEGAL4-1=AMPCR
ILLEGAL4-1=AMPCR

OPR74XX:
OPR74X0-1=AMPCR
OPR74X1-1=AMPCR
OPR74X2-1=AMPCR
OPR74X3-1=AMPCR
OPR74X4-1=AMPCR
OPR74X5-1=AMPCR

```

```

01180000 0
01190000 0
01200000 0
01210000 0
01220000 0
01230000 0
01240000 0
01250000 0
01260000 0
01270000 0
01280000 0
01290000 0
01300000 0
01310000 0
01320000 0
01330000 0
01340000 0
01350000 0
01360000 0
01370000 0
01380000 0
01390000 0
01400000 0
01410000 0
01420000 0
01430000 0
01440000 0
01450000 0
01460000 0
01470000 0
01480000 0
01490000 0
01500000 0
01510000 0
01520000 0
01530000 0
01540000 0
01550000 0
01560000 0
01570000 0
01580000 0
01590000 0
01600000 0
01610000 0
01620000 0
01630000 0
01640000 0
01650000 0
01660000 0
01670000 0
01680000 0
01690000 0
01700000 0
01710000 0
01720000 0
01730000 0
01740000 0
01750000 0
01760000 0
01770000 0

%PRINT MESSAGE & RETURN TO HALF FETCH
%PRINT MESSAGE & RETURN TO HALF FETCH
%PRINT MESSAGE & RETURN TO HALF FETCH
%PRINT MESSAGE & RETURN TO HALF FETCH

```



```

0096 096C 0000 0030 00C0
0097 097C 0000 0000 00CC
0098 0980 00C0 0030 00C0
0099 0990 00C0 0030 00C0
009A 09A0 00C0 0000 0040
009B 09B0 0000 0030 0040
009C 09C0 0000 0030 00C0
009D 09D0 0000 0030 00C0
009E 09E0 0000 0030 00C0
009F 09F0 00C0 0030 0040

0CA0 480C 0000 0020 C0F0
0CA1 0988 00C0 0000 00FC
0CA2 0A0C 00C0 0030 C060
0CA3 48C9 00C1 0830 00F0
0CA4 A00C 00C0 0030 0020
0CA5 4809 C040 2030 00F0

00A6 0A1C 00C0 0030 0060
00A7 380B 00C0 0030 00F0
00A8 0A20 0000 0030 0060
00A9 180B 0000 0030 00F0
0CAA 0A30 00C0 0030 C060

0CAB 4809 C0C0 001C 00FC
0CAC AB09 0000 0030 C0F0
0CA0 AC08 C0C0 2C00 00F0

0CAF 4809 0C40 4000 00F0
0CB0 0000 00C0 0000 0020
0CB1 4809 0C40 0030 00F0
0CB2 580B 0000 0030 C0F0
0CB3 0A4C 0000 0030 C060

0CB4 4809 A0C0 8800 00F0
0CB5 1070 0000 0000 00A0
0CB6 4809 2C56 0830 00FC
0CB7 4809 A0C0 8040 00F0
0CB8 2080 0000 0000 C0B0
0CB9 4809 0000 0030 00F0
0CBA 2824 2C40 100C 00F0
0CBB 382D 0000 0000 00F0

0CBC 9809 0000 0030 18FC
0CB0 4809 A0C1 0000 C0F0
0CBE 0000 0000 0000 0020
0CBF 4809 A0C0 9030 00F0
0CC0 200C 00C0 0030 0030
0CC1 4809 C001 8830 40FC
0CC2 8000 0000 0000 C020

OPR74X6-1=AMPCR
OPR74X7-1=AMPCR
OPR77XX:
OPR77X0-1=AMPCR
OPR77X1-1=AMPCR
ILLEGAL4-1=HPCR
ILLEGAL4-1=HPCR
OPR77X4-1=AMPCR
OPR77X5-1=AMPCR
OPR77X6-1=AMPCR
ILLEGAL4-1=HPCR

% START BY TRYING TO SET GC1 WHICH MEANS THE
% LOADER HAS LOADED THE OBJECT CODE AND
% PRESET THE REGISTERS
% IN GENERAL A2 CONTAINS ASR/PAR, A1 CONTAINS CURRENT INSTRUCTION
START: 0 = BR1; WAIT
SET GC1; WHEN GC1 THEN STEP
GETPAR-1 = CPCR
1 L=B; RESET GC2
COMP 10-SAR
A2+B=A2

IFETCH:
PUTPAR-1 = CPCR
IF 1R0 THEN STEP ELSE SKIP
100HPRG-1 = CPCR
IF GC2 THEN STEP ELSE SKIP
GC2CHECK-1 = CPCR

IFETCH1:
A2 = MAR2
MR2; IF ROC
WHEN ROC THEN A2 + 1 = A2,BEX % INC PAR
IFETCHRENOTE: XENTRY FROM EXECUTE REMOTE OPR02X2
B = A1
B R = B
16 = SAR
B
IF 1ST THEN STEP ELSE SKIP
INDIRECT-1 = CPCR
IFETCH2:
A1 R = D
7 = LIT; 17 = SAR
LIT AND B = B
% E-FIELD
A1 R = AMPCR
% OPCODE
BBASE = LIT; 26 = SAR
STEP
LIT + B = MAR,A3; EXEC; IF LC1 % BCB ADDR =>MAR,A3
JUMP; IF LC2
% JUMP TO OPCODE

% ***** SUBR FOR MEMORY FETCHES *****
HALFFETCH: % FETCH THE NEXT HALFWORD INSTRUCTION
MW1; IF SAI
% WRITE RESULTS OF PREVIOUS HALFWORD
% SOME INSTRUCTIONS ENTER HERE AT HALFFETCH+1
A1 C=A1
16-SAR
A1 R=A3
26-SAR
A2 C=B,CSAR
20-SAR
% GET HALFWORD FLAG TO LSR

```


| | | | | | | | | |
|------|------|------|------|------|--|------------------------------------|----------|---|
| 00C3 | 4809 | 0C40 | 003C | 00F0 | B | TEST BIT SET=>PREVIOUS HALFWORD | 02380C00 | D |
| CCC4 | 5C08 | 0C01 | A000 | 00F0 | IF LST THEN BIT0 C=A2; STEP ELSE SKIP | %SET BIT IF CLEARED | 02390C00 | D |
| 00C5 | 0A50 | 0000 | 0000 | 0040 | IFETCH-1=MPCR | | 02400000 | D |
| 00C6 | 3809 | 0C01 | A030 | 00FC | BIT1 C=A2; IF LC2 | %CLEAR REPEAT BIT IF SET; GET NEXT | 02410C00 | D |
| 00C7 | 8019 | 0000 | 0000 | 00F0 | IF NOT IRO THEN SKIP | | 02420C00 | D |
| 00C8 | 0A5C | 0003 | 0030 | 0060 | PUTPAR-1=CPCR | | 02430C00 | D |
| 00C9 | 8019 | 0000 | 0030 | 00F0 | IF NOT IRO THEN SKIP | | 02440C00 | D |
| 00CA | 0A6C | 0000 | 0030 | 0060 | 100MPREG-1=CPCR | | 02450C00 | D |
| 00CB | 4809 | EC00 | 0030 | 00F0 | A3=AMPCR | | 02460C00 | D |
| 00CC | 4809 | 0000 | 0030 | 00F0 | STEP | | 02470C00 | D |
| 00CD | 2824 | E012 | 0C00 | 00F0 | A3 EOL 0; EXEC; IF LC1 | %IF LOWER OPC=C DON'T EXECUTE | 02480C00 | D |
| 00CE | 6029 | 0000 | 0030 | 00F0 | IF FALSE THEN JUMP | | 02490C00 | D |
| 00CF | 08C0 | 0003 | 0C30 | 0040 | HALF FETCH=MPCR | | 02500000 | D |
| 00D0 | A809 | A000 | 9030 | 00F0 | IFETCH: %MAR=B(8) ADDRESS; RETURN MAR2=Y-ADDR; MIR=Y(K) | | 02510C00 | D |
| 00D1 | 807C | 0003 | 0030 | 0040 | A1 R=A3; IF RDC | | 02520C00 | D |
| 00D2 | A809 | E156 | 1C35 | 08F0 | 7=LIT; 20=SA | %ISOLATE K FIELD | 02530000 | C |
| 00D3 | 1019 | 0000 | 0030 | 00F0 | A3 AND LIT=CTR; A3; MIR; IF RDC | %SAVE K IN CTR; READ F(B) | 02540C00 | D |
| 00D4 | 0A70 | 0000 | 0C00 | 0040 | IF NOT GC2 THEN SKIP | %GC2 SET IF SPECIAL CONDITIONS | 02550C00 | D |
| 00D5 | AC08 | 0000 | 0C30 | 00F0 | YABNORMAL-1=MPCR | %CHECK FOR SPECIAL CONDITIONS | 02560C00 | D |
| 00D6 | 4809 | E002 | 0000 | 00F0 | WHEN RDC THEN BEX | %B=B(8) | 02570C00 | D |
| 00D7 | 6019 | 0C40 | 0030 | 00F0 | YREFETCH: NOT A3 | | 02580C00 | D |
| 00D8 | 0A8C | 0000 | 0C30 | 0040 | IF ABT THEN B=MIR; SKIP | %CHECK K=0 | 02590C00 | D |
| 00D9 | 4809 | 0018 | 8830 | 00F0 | KNOTZERO-1=MPCR | %JUMP HERE IF KNOT = ZERO | 02600C00 | D |
| 00DA | 0000 | 0000 | 0000 | 0020 | B111 R=B | %ARRIVE HERE FROM ABNORMAL NOT CA | 02610C00 | D |
| 00DB | 4809 | A0C1 | 1C30 | 00F0 | 16=SA | | 02620000 | D |
| 00DC | 4809 | EC00 | 0000 | 00F0 | A1 L=A3 | %CHECK SIGN OF SY-FIELD | 02630C00 | D |
| 00DD | 4C19 | EC53 | 9000 | 00F0 | IF MST THEN A3 OR B C=A3; B; MIR; SKIP | %A3= SY SHIFTED LEFT TO MSB | 02640C00 | D |
| 00DE | 4809 | E000 | 9000 | 00F0 | A3 R = A3; B; MIR | %EXTEND NEGATIVE SIGN | 02650C00 | D |
| 00DF | 482D | EC43 | 1C30 | 00F0 | A3*B=A3; MIR; JUMP | %EXTEND POSITIVE SIGN | 02660C00 | D |
| 00E0 | 4809 | A003 | 9000 | 00F0 | KNOTZERO: A1 R=A3 | %Y=SY(SE)+F(B); GO TO OFCODE | 02670C00 | D |
| 00E1 | 907C | 0000 | 0000 | 0090 | 7=LIT; 13=SA | %ISOLATE S-FIELD | 02680C00 | D |
| 00E2 | 4809 | E156 | 1C30 | 00F0 | A3 AND LIT=A3 | | 02690C00 | D |
| 00E3 | 4809 | E140 | 0C30 | 00F0 | A3*IT=WAR | %MAR= ADDRESS OF S(S) | 02700C00 | D |
| 00E4 | 0100 | 0000 | 0030 | 00E0 | SRASE=LIT | | 02710C00 | D |
| 00E5 | A809 | A0C1 | 1C30 | 48FC | A1 L=A3; C; SAR; MIR; IF RDC | %READ S(S) | 02720000 | C |
| 00E6 | 4809 | EC00 | 9030 | 00F0 | A3-R=A3 | %A3=Y-FIELD ISOLATED RIGHT | 02730C00 | D |
| 00E7 | AC08 | EC40 | 1C00 | 00F0 | WHEN RDC THEN A3+B=A3; BEX | %B=S(S); A3=Y+B(B) | 02740C00 | D |
| 00E8 | 4809 | EC43 | 001C | 00F0 | A3*B=MAR2 | | 02750C00 | D |
| 00E9 | 4809 | 0640 | 0C30 | 24F0 | IFETCH1: %RETURN HERE FROM Y-ABNORMAL UNLESS K=0 OR CHAR ADDR. | | 02760C00 | D |
| 00EA | 4809 | 08C2 | 9000 | 00F0 | AMPCR=MIR; ASE | %SELECT BR2=K; SAVE RETURN | 02770C00 | D |
| 00EB | 0000 | 0000 | 0000 | 0030 | NOT CTR R=A3 | %RESTORE K IN MSB INPUT FROM CTR | 02780C00 | D |
| 00EC | 4809 | EC40 | 0040 | 00F0 | 24=SA | | 02790C00 | D |
| 00ED | 0A90 | 0000 | 0030 | 00CC | A3*AMPCR=AMPCR | %SHIFT CTR TO LSB | 02800C00 | D |
| 00EE | 4809 | 0000 | 0030 | 00F0 | KTABE-1=AMPCR | %SET UP K-HANDLER ROUTINES | 02810C00 | D |
| 00EF | 4824 | 0000 | 0030 | 00F0 | MIR; IF RDC | %READ Y VALUE | 02820C00 | D |
| 00F0 | 4836 | 0000 | 0000 | 00F0 | EXEC | %EXECUTE K-HANDLER ROUTINE | 02830C00 | D |
| 00F1 | 4809 | 0C40 | 0040 | 00FC | CALL | | 02840C00 | D |
| 00F2 | 4809 | 0C00 | 0030 | 00FC | B = AMPCR | | 02850C00 | D |
| 00F3 | 482D | 0000 | 0030 | 00F0 | STEP | | 02860C00 | D |
| 00F4 | 480C | 0000 | 0000 | 00F0 | JUMP | | 02870C00 | D |
| 00F5 | 0A4C | 0000 | 0C00 | 00C0 | KTABE: WAIT | | 02880C00 | D |
| 00F6 | 0A8C | 0003 | 0C30 | 00C0 | KR1-1=AMPCR, KR2-1=AMPCR, KR3-1=AMPCR, KR4-1=AMPCR | | 02890C00 | D |
| 00F7 | 0A8C | 0000 | 0030 | 00C0 | | | 02900C00 | D |
| 00F8 | 0A00 | 0000 | 0000 | 00C0 | | | 02910C00 | D |
| 00F9 | 0AEO | 0000 | 0000 | 00C0 | | | | |
| 00FA | 0AFO | 0000 | 0000 | 00C0 | | | | |
| 00FB | 080C | 0000 | 0000 | 00C0 | | | | |
| | | | | | KR5-1=AMPCR, KR6-1=AMPCR, KR7-1=AMPCR | | 02920C00 | D |


```

0198 4809 0C40 9000 00F0
0199 81F0 0C00 0C00 00A0
019A 4809 E156 4000 00F0
019B 4809 E0C0 9000 00F0
019C 9000 0000 0000 0000
019D 4809 E156 0800 80F0
019E 4809 AC5E 4028 40F0
019F 0180 0000 0000 00E0
01A0 7409 A0DE 4000 00F0
01A1 4FC9 2C5E 4000 00F0
01A2 7409 A0DE 4000 00F0
01A3 4809 0C19 900C C8FC
01A4 AC08 0000 0000 C0F0
01A5 2F09 0C46 001C 00F0
01A6 4809 0C40 001C C0F0
01A7 4809 A000 0000 8CF0
01A8 AC08 EC56 0C30 00F0
01A9 4809 EC44 0F00 C0F0
01AA 4809 0C41 8030 C0F0
01AB 9809 0000 0C00 1CF0
01AC 9C08 A0C1 0098 00F0
01AD 81AC 0000 0000 0090
01AE 4809 2001 1020 C8F0
01AF 01FC 0000 0000 C0E0
01B0 AC08 0000 0C38 C0F0
01B1 01CC 0000 0000 00E0
01B2 4809 EC56 0F0C 08F0
01B3 AC08 0000 0C30 00F0
01B4 4809 0C40 001C 00F0
01B5 9809 0000 0C00 1CF0
01B6 9828 0000 0000 00F0
01B7 480C 0000 0000 00F0
01B8 0BAC 0000 0000 00C0
01B9 0880 0000 0000 00C0
01BA 08CC 0000 0C00 00CC
01BB 08DC 0000 0000 00C0
01BC 08EC 0000 0C00 C0C0
01BD 08FC 0000 0C00 00C0
01BE 0C00 0000 0000 00C0

C1BF AC08 0000 0C00 C0F0
01C0 4809 0C40 8800 00F0
01C1 000C 0000 0000 0020
01C2 4809 0C41 1020 C0F0
01C3 4809 0C41 0800 00F0
01C4 4809 0C40 8800 00F0
01C5 4809 EC5C 0030 00F0
01C6 0C10 0000 0000 0040

01C7 AC08 0000 0C00 00F0
01C8 4809 0C41 0800 00F0
01C9 0000 0000 0C00 0020
01CA 4809 0C40 9000 00F0
01CB 4809 0C41 0800 00FC
01CC 4809 EC5C 1000 C0FC
01CD 0C20 0000 0000 0040

01CE AC08 0000 0C00 00F0
01CF 0C30 0000 0000 C040

0198 4809 0C40 9000 00F0
0199 81F0 0C00 0C00 00A0
019A 4809 E156 4000 00F0
019B 4809 E0C0 9000 00F0
019C 9000 0000 0000 0000
019D 4809 E156 0800 80F0
019E 4809 AC5E 4028 40F0
019F 0180 0000 0000 00E0
01A0 7409 A0DE 4000 00F0
01A1 4FC9 2C5E 4000 00F0
01A2 7409 A0DE 4000 00F0
01A3 4809 0C19 900C C8FC
01A4 AC08 0000 0000 C0F0
01A5 2F09 0C46 001C 00F0
01A6 4809 0C40 001C C0F0
01A7 4809 A000 0000 8CF0
01A8 AC08 EC56 0C30 00F0
01A9 4809 EC44 0F00 C0F0
01AA 4809 0C41 8030 C0F0
01AB 9809 0000 0C00 1CF0
01AC 9C08 A0C1 0098 00F0
01AD 81AC 0000 0000 0090
01AE 4809 2001 1020 C8F0
01AF 01FC 0000 0000 C0E0
01B0 AC08 0000 0C38 C0F0
01B1 01CC 0000 0000 00E0
01B2 4809 EC56 0F0C 08F0
01B3 AC08 0000 0C30 00F0
01B4 4809 0C40 001C 00F0
01B5 9809 0000 0C00 1CF0
01B6 9828 0000 0000 00F0
01B7 480C 0000 0000 00F0
01B8 0BAC 0000 0000 00C0
01B9 0880 0000 0000 00C0
01BA 08CC 0000 0C00 00CC
01BB 08DC 0000 0000 00C0
01BC 08EC 0000 0C00 C0C0
01BD 08FC 0000 0C00 00C0
01BE 0C00 0000 0000 00C0

C507 0C0C 0
0508 0C00 0
0509 0C00 0
0510 0C00 0
0511 0C00 0
0512 0C00 0
0513 0C00 0
0514 0C00 0
0515 0C00 0
0516 0C00 0
0517 0C0C 0
0518 0C00 0
0519 0C00 0
0520 0C00 0
0521 0C00 0
0522 0C00 0
0523 0C00 0
0524 0C00 0
0525 0C00 0
0526 0C00 0
0527 0C00 0

B R=A3
20=SAR;J31=LIT
A3 AND LIT=A1
A3 R=A3
5=SAR
A3 AND LIT=B,SAR
A1-B=A1,CSAR,LHAR
YADDR=LIT
IF NOT ADV THEN A1-1=A1
IF HST THEN LIT-B=A1;SET LC1 XA1-32-W=P
IF NOT ADV THEN A1-1=A1
B111 R=A3;HR1;IF RDC
WHEN RDC THEN BEX
IF LC1 THEN B+1=HAR2,BHI;SET LC1;SKIP
B=HAR2,BHI
A1=SAR;HR2;IF RDC
WHEN RDC THEN A3 AND B=HIR,BEX XE=Y) SHIFTED BY P
A3 NRI B=EBI
B C=HIR
HW2;IF SAI
WHEN SAI THEN A1 L=HIR,LHAR XE LEFT JUSTIFIED TO POSITION
IAN=LIT;COMP 20=SAR
LIT L=A3;HR1;IF RDC
31=LIT
WHEN RDC THEN BEX,LHAR
IANADDR=LIT
A3 AND B=EBI;HR1;IF RDC
WHEN RDC THEN B=HIR,BEX
B=HAR2
HW2;IF SAI
WHEN SAI THEN JUMP
KSTABLE: WAIT
KS1-1=AMPCR. KS2-1=AMPCR. KS3-1=AMPCR
KS4-1=AMPCR. KS5-1=AMPCR. KS6-1=AMPCR. KS7-1=AMPCR

KS1: X(A 15-0)=>Y 15-0
WHEN RDC THEN BEX
B R=B
16 = SAR
B L=A3,BMI
B L=B
B R=B
A3 OR B=HIR
OUTPUT2NORET-1=HPCR
X(A15-0)=>Y 31-16
WHEN RDC THEN BEX
B L=B
16 = SAR
B R=A3,BMI
B L=B
A3 OR B=A3,HIR
OUTPUT2NORET-1=HPCR
X(A31-0)=>Y(31-0)
WHEN RDC THEN BEX
OUTPUT2NORET-1=HPCR
X(A 7-0)=>Y 7-0

KS2: X(Y31-16) UNCHANGED
XA3=Y31-16
X(A 15-0)=>Y 15-0
XY 15-0 UNCHANGED
XA3=Y 15-C

KS3:
KS4:

```



```

0100 AC08 00C0 0C30 C0F0
0101 0000 0000 0030 0010
0102 4809 0C40 8800 40F0
0103 4809 0C41 1030 40F0
0104 4809 0C41 0800 40F0
0105 4809 0C40 8800 00F0
0106 4809 EC5C 1030 00F0
0107 0C4C 00C0 0C30 0040

0108 AC08 20C1 1C30 40F0
0109 0FF0 0000 0000 0080
010A 4809 EC44 1030 00F0
010B 4809 0C41 0830 00F0
010C 4809 0C40 8830 00F0
010D 0000 00C0 0030 C020
010E 4809 EC5C 1030 00F0
010F 0C5C 00C0 0C30 0040

01E0 AC08 2001 1C30 00F0
01E1 0FF0 0000 0000 00A0
01E2 4809 EC44 1000 C0F0
01E3 4809 0C41 0800 00F0
01E4 000C 00C0 0000 0010
01E5 4809 0C40 8830 00F0
01E6 4809 EC5C 1C30 00F0
01E7 0C6C 0000 0C00 0040

01E8 AC08 0000 0C0C 00F0
01E9 4809 0C41 0830 40F0
01EA 0000 0000 0C30 C030
01EB 4809 0C40 9000 00F0
01EC 4809 0C41 0830 00FC
01ED 4809 EC5C 1C30 00F0
01EE 0C7C 0000 0070 0040

01EF 4809 0000 0C30 00F0
01F0 0C8C 0000 0000 0040

01F1 4809 03C1 1030 00F0
01F2 0000 00C0 0030 C020
01F3 4809 EF5C 0C30 00F0
01F4 0C90 0000 0000 0040

01F5 4809 2001 1C00 00F0
01F6 0020 0000 0000 C0A0
01F7 4809 EC5C 0030 00F0
01F8 0C40 0000 0030 0040

01F9 4809 20C1 1030 00F0
01FA 0070 0000 0000 C0A0
01FB 4809 EC5C 0C30 00F0
01FC 0C8C 00C0 0030 0040

WHEN RDC THEN BEX
B=SAR
B R=B,CSAR
B L=A3,BN1,CSAR
B L=B,CSAR
B R=B
A3 OR B=A3,MIR
OUTPUT2NORET-1=MPCR
KS5: X(A7-0)>Y15-8
WHEN RDC THEN LIT L = A3,CSAR,BEX
255 = LIT; COMP 8 = SAR
A3 NR1 B=A3,BN1
B L=B
B R=B
16=SAR
A3 OR B=A3,MIR
OUTPUT2NORET-1=MPCR
KS6: X(A7-0)>Y23-16
WHEN RDC THEN LIT L=A3,BEX
255=LIT;16=SAR
A3 NR1 B=A3,BN1
B L = B
COMP 24 = SAR
B R=B
A3 OR B=A3,MIR
OUTPUT2NORET-1=MPCR
KS7: X(A7-0)>Y31-24
WHEN RDC THEN BEX
B L=B,CSAR
COMP 8 = SAR
B R=A3,BN1
B L = B
A3 OR B=A3,MIR
OUTPUT2NORET-1=MPCR

X
X
X ***** SUBR. FOR I/O FUNCTIONS *****
I0DUMPREG: *****
0 = MIR
SIGLOADER-1=MPCR
I0DUMPADR: XB=ADDRESS TO PRINT
1 L = A3
COMP 16 = SAR
A3 OR BMAR=MIR
SIGLOADER-1=MPCR
CARDREAD: XREAD A CARD INTO ADDRESS IN B
LIT L=A3
2=LIT;COMP 16=SAR
A3 OR B=MIR
SIGLOADER1-1=MPCR
PRINTLINE: XPRINT AN OUTPUT BUFFER ADDRESSED IN B
LIT L=A3
7=LIT; COMP 16=SAR
A3 OR B=MIR
SIGLOADER1-1=MPCR
DISKREAD: XREAD/WRITE DISK USING INFO AT Y+(B)+(S)
XB=NUMBER OF WORDS
XB*1=DISK SECTOR TO BE ACCESSED
XB*2= 3=>READ; 4=>WRITE
XB*3=ADDRESS OF BUFFER

```



```

01FD 4809 0C43 001C 00F0
01FE 4809 0000 0000 00F0
01FF AC08 0F46 0C1C 00F0
0200 4809 0C41 1050 00F0
0201 0000 0000 0000 0020
0202 AC08 0F46 0C1C 00F0
0203 4809 EC5C 1000 00F0
0204 AC08 0F46 0C1C 00F0
0205 4809 0C41 4030 00F0
0206 AC08 0008 001C 00F0
0207 4809 E000 0050 00F0
0208 4809 AC5C 4030 10F0
0209 9C08 ACC0 0050 00F0
020A 0A5C 00C0 0030 00C0

020B 4809 0013 0010 00F0
020C 4809 0000 0000 10F0
020D 1ACB 00C0 0000 00F0
020E 49C9 0000 0000 00F0
020F 9848 00C0 0030 00F0
0210 8AC8 0000 0030 00F0
0211 0988 0000 0000 00F0
0212 2800 0000 0030 00F0
0213 18A8 0000 0000 00F0

0214 4809 0C41 0800 00F0
0215 005C 0000 0030 00AC
0216 4809 2C5D 8080 00F0
0217 20AC 0000 0030 0060
0218 4809 C0C1 0830 00F0
0219 0060 0000 0030 00E0
021A 3819 2C5D 8050 00F0
021B 2090 00C0 0030 0040
021C 08C0 0000 0030 00C0
021D 20A0 0000 0030 0040
021E 481E 0000 0800 00F0
021F 4849 0000 0830 00F0
0220 2130 0000 0030 0040
0221 4858 2000 0800 00F0
0222 4809 20C0 0830 00F0
0223 0030 00C0 0000 00E0
0224 213C 00C0 0030 0040
0225 4818 00C0 0800 00F0
0226 4849 00C0 0800 00F0
0227 2130 0000 0030 0040

0228 4809 C0C0 0030 00F0
0229 010C 0000 0000 00E0
022A 00CC 00C0 0030 0040

022B 4809 0000 0030 00F0
022C 010C 00C0 0000 00E0
022D 4809 0000 0030 08F0
022E AC08 0000 00C0 00F0

B=MAR2
MR2,IF RDC
WHEN RDC THEN BMAR+1=MR2,BEX
B L=A3,MR2,IF RDC      %A3=NUM WORDS/0
COMP 16=5AR
WHEN RDC THEN BMAR+1=MR2,BEX
A3 OR B=A3,MR2,IF RDC  %A3=NUM WORDS/SECTOR
WHEN RDC THEN BMAR+1=MR2,BEX
B L=A1,MR2,IF RDC      %A1=3 OR 4/0
WHEN RDC THEN B11C=MR2
A3=MIR
A1 OR B=A1,MR2,IF SAI  %A1=3-4/ADDRESS
WHEN SAI THEN A1=MIR
SIGLOADER1: IFETCH-1=AMPCR
SIGLOADER:
B111=MR2,CTR
MR2,IF SAI
IF GC2 THEN RESET GC2, STEP ELSE SKIP
SET LC1                %SIGNALS GC2 MUST BE RESET
WHEN SAI THEN RESET GC1
INC,WHEN COV THEN RESET GC2,STEP
SET GC1,WHEN GC1 THEN STEP
IF LC1 THEN STEP ELSE JUMP
SET GC2, WHEN GC2 THEN JUMP

%
ERRORMSG: %B=NUMBER OF ERROR MESSAGE TO BE PRINTED
%0=ILLEGAL INSTRUCTION
%1=INSTRUCTION NOT IMPLEMENTED
%2=SQUARE ROOT OF NEGATIVE NUMBER
%PRINT ERROR MESSAGE
B L=B
16=5AR,5=LIT
LIT OR B C=MIR
SIGLOADER-1=CPCR
A2 L=B
6=LIT
LIT OR B C=MIR,IF LC2 THEN SKIP
%OUTPUT ADDRESS AND CONTENTS
%RETURNS TO IFETCH
SIGLOADER1-1=MPCR
HALFFETCH=AMPCR
SIGLOADER-1=MPCR
ILLEGAL: 0=B,SKIP
ILLEGAL4: 0=B,SET LC2
ERRORMSG-1=MPCR
ILLEGALCMR1: LIT=B,SET LC2,SKIP
ILLEGALCMR: LIT=B
3=LIT
ERRORMSG-1=MPCR
UNWRITTEN: 1=B,SKIP
UNWRITTEN4: 1=B,SET LC2
ERRORMSG-1=MPCR
%
% ***** SUBPR. FOR REGISTER STORES *****
PUTPAR: A2=MIR,LMAR
PAR=LIT
OUTPUT1-1=MPCR
GETPAR: LMAR
PAR=LIT
MR1,IF RDC
WHEN RDC THEN BEX

05880C00 D
05890C00 D
05900C00 D
05910C00 D
05920C00 D
05930000 D
05940C00 D
05950C00 D
05960C00 D
05970C00 D
05980C00 D
05990C00 D
06000C00 D
06010C00 D
06020C00 D
06030C00 D
06040C00 D
06050C00 D
06060C00 D
06070C00 D
06080C00 C
06090C00 D
06100C00 D
06110C00 D
06120C00 D
06130C00 D
06140C00 D
06150C00 D
06160C00 D
06170C00 D
06180C00 D
06190C00 D
06200C00 D
06210C00 D
06220C00 D
06230C00 D
06240C00 D
06250C00 D
06260C00 D
06270C00 D
06280C00 D
06290C00 D
06300C00 D
06310C00 D
06320C00 D
06330C00 D
06340C00 D
06350C00 D
06360C00 D
06370C00 D
06380C00 D
06390C00 D
06400C00 D
06410C00 C
06420000 D
06430C00 D
06440000 D
06450C00 D
06460C00 D
06470C00 D

```



```

022F 4820 0040 2000 00F0
0230 0A50 0000 0000 00C0
0231 9809 0000 0000 18F0
0232 9C28 0000 0000 00F0
0233 0A50 0000 0000 00C0
0234 9809 0000 0000 1CF0
0235 9C08 0000 0000 00F0
0236 8800 0000 0000 00F0
0237 1F0C 0000 0000 0040

0238 1C19 C000 9000 00F0
0239 2320 0000 0000 0040
023A 8180 0000 0000 0040
023B 4809 E000 0000 00F0
023C 5C19 0000 0000 00F0
023D 2320 0000 0000 0040
023E 4809 0000 0000 C8F0
023F AC08 0000 0000 00F0
0240 3C09 0040 00C0 00F0
0241 2320 00C0 0000 0040

0242 2A59 0000 0000 00F0
0243 22F0 0000 0000 0040
0244 230C 0000 0000 0060
0245 4809 A000 001C 00F0
0246 2370 0000 0000 004C
0247 4809 A000 9000 00F0
0248 8070 0000 0000 00A0
0249 4820 E156 1000 00F0
024A 4809 A000 8000 00F0
024B 8070 0000 0000 00A0
024C 4820 2C56 080C 00F0
024D 4809 A000 9000 00F0
024E 8070 0000 0000 00A0
024F 4820 E156 100C 00F0

0250 4809 A000 9000 00F0
0251 9030 0000 0000 00A0
0252 4820 E156 1000 00F0

0253 4809 E15E 0000 00F0
0254 010C 0000 0000 00EC
0255 73E9 0000 0000 00F0
0256 4809 E15E 0000 00F0
0257 0180 0000 0000 00EC
0258 7C19 E15E 0000 00F0
0259 28ED 0000 0000 00F0
025A 0300 0000 0000 00EC
025B 7C19 E15E 0000 00F0
025C 2820 0000 0000 00F0
025D 0580 0000 0000 00EC
025E 7C19 E15E 0000 00F0
025F 28ED 0000 0000 00F0
0260 060C 0000 0000 00EC
0261 7C19 E15E 0000 00F0
0262 2820 0000 0000 00F0
0263 280C 0000 0000 0090
0264 7019 0000 0000 00F0

```

```

B=A2;JUMP
OUTPUT:  MNO RETURN
IFEICH-1=AMPCR
OUTPUT1:  HW1;IF SAI
WHEN SAI THEN 0;JUMP
OUTPUT2NORET:  IFEICH-1=AMPCR
OUTPUT2:  HW2;IF SAI
WHEN SAI THEN 0;STEP
IF IR0 THEN STEP ELSE JUMP
100MADR-1=MPCR
YOUT:  XY-ADDRESS IN MAR2 CAME FROM FRECALCULATED EXCEPT REP
IF G02 THEN A2 R=A3;SKIP  XREPLACE REPEAT IS SPECIAL
OUTPUT2NORET-1=MPCR
20=SAR;YADDR=LIT
A3
IF LST THEN LMAR;SKIP
OUTPUT2NORET-1=MPCR
HW1;IF ROC
WHEN ROC THEN BEX
IF L02 THEN B=MAR2
OUTPUT2NORET-1=MPCR
X ***** LOGICAL SUBROUTINES *****
OUTLOGIC:  XY ADDRESS IN B,A ADDR IN MAR1
IF L01 THEN SET LC2;SKIP
OUTPUT-1=MPCR
OUTPUT1-1=OPCR
A1=MAR2
YOUT-1=MPCR
GETF:  A1 R=A3
20=SAR;7=LIT
A3 AND LIT=A3;JUMP
GETAB:  A1 R=B
23=SAR;7=LIT
LIT AND B=B;MAR;JUMP
GETAA3:  A1 R=A3
23=SAR;7=LIT
A3 AND LIT=A3;MAR;JUMP
GETF1:  A1 R=A3
21=SAR;3=LIT
A3 AND LIT=A3;JUMP
CHECKCHR:  XCHR ADDRESS IN A3;L01 SET IF PRIVILEGED;RETURN LC1 IF LEGAL
A3 LSS LIT
16=LIT
IF TRUE THEN SET L01; JUMP
A3 LSS LIT
24=LIT
IF FALSE THEN A3 LSS LIT; SKIP
JUMP; IF L01 THEN SET LC1
48=LIT
IF FALSE THEN A3 LSS LIT;SKIP
JUMP; IF L01
88=LIT
IF FALSE THEN A3 LSS LIT;SKIP
JUMP; IF L01 THEN SET LC1
96=LIT
IF FALSE THEN A3 LSS LIT;SKIP
JUMP; IF L01
128=LIT;10=SAR
IF TRUE THEN SKIP
06480000 0
06490000 0
06500000 0
06510000 0
06520000 0
06530000 0
06540000 0
06550000 0
06560000 0
06570000 0
06580000 0
06590000 0
06600000 0
06610000 0
06620000 0
06630000 0
06640000 0
06650000 0
06660000 0
06670000 0
06680000 0
06690000 0
06700000 0
06710000 0
06720000 0
06730000 0
06740000 0
06750000 0
06760000 0
06770000 0
06780000 0
06790000 0
06800000 0
06810000 0
06820000 0
06830000 0
06840000 0
06850000 0
06860000 0
06870000 0
06880000 0
06890000 0
06900000 0
06910000 0
06920000 0
06930000 0
06940000 0
06950000 0
06960000 0
06970000 0
06980000 0
06990000 0
07000000 0
07010000 0
07020000 0
07030000 0
07040000 0
07050000 0
07060000 0
07070000 0

```



```

0265 282D 00C0 0C00 00F0
0266 28ED 0000 0C00 00FC

0267 2019 C000 A000 C0F0
0268 0A50 0000 0000 0040
0269 0000 0000 0000 0020
026A 3809 C001 0B00 00F0
026B 4809 EC5C 2000 00F0
026C 2270 0000 0C00 0060
026D 0A50 00C0 0000 0040

026E 4809 A0C1 1000 00F0
026F 2000 0C00 0000 0030
0270 4809 E000 0000 00F0
0271 A000 00C0 0000 0020
0272 4809 E156 8000 00FC
0273 10E0 0000 0000 0080
0274 4809 E156 8800 00F0
0275 3600 0000 0000 0080
0276 4809 2052 0C00 20FC
0277 0020 0000 0000 00E0
0278 6FC9 2F40 0000 00F0
0279 0080 0000 0000 C0E0
027A 4809 2C52 0000 00F0
027B 0030 0000 0000 00E0
027C 68C9 0000 0000 00F0
027D 2A08 0000 0000 C8F0
027E AC18 0000 0000 00F0
027F 4809 E156 0800 00FC
0280 83F0 0000 0000 0080
0281 4809 E000 8000 00F0
0282 4809 2C56 1000 88F0
0283 4809 E15E 0000 00F0
0284 0200 0000 0000 C0E0
0285 7FC9 E15E 0000 80F0
0286 AC28 00C0 0C00 00F0

0287 0BCC 00C0 0C00 C0C0
0288 280B 0000 0000 00FC
0289 0A50 0000 0000 00C0

028A 4849 EC52 0000 00F0
028B 6C19 0B1D A000 40FC
028C 4809 080F AC00 40F0

028D 8000 0000 0C00 0030
028E 4809 EC4C 0C00 00F0
028F 480B EC5E 0000 C0F0
0290 7C2B 080F AC00 00F0
0291 7A29 08CF A000 00F0
0292 482D 0B1D A000 C0FC

JUMP, IF LC1
JUMP, IF LC1 THEN SET LC1
***** JUMP SUBROUTINES *****
PUTJUMP: XA3=JUMP ADDRESS;STORE IN A34PAR
IF LC1 THEN A2 R=A2;SKIP
IFETCH-1=MPCR
16=5AR
A2 L=8;IF LC2
A3 OR B=A2
PUTPAR-1=CPCR
IFETCH-1=MPCR

X
X
X ***** SHIFT SUBROUTINES *****
SHIFTAMOUNT: XRETURNS B=(A-REG);MAR=A-ADDR;SAR=SHIFTAMOUNT
A1 L=A3
COMP 6=5AR
A3 R=A3
22=5AR
A3 AND LIT R=MAR
14=LIT;I=5AR
A3 AND LIT R=B
96=LIT;I=5=5AR
LIT EOL B;ASR
2=LIT
IF TRUE THEN LIT+6MAR=MAR;SET LC1
BBASE=LIT
LIT EOL B;IF RDC
3=LIT
IF TRUE THEN SET LC1
IF LC1 THEN MR;STEP ELSE SKIP
WHEN RDC THEN BEX;SKIP
A3 AND LIT=B
63=LIT;I=5AR
A3 R=MAR;CTR
LIT AND B=5AR;A3;MR;IF RDC
A3 GE0 LIT
32=LIT
IF TRUE THEN A3-LIT=5AR;SET LC1
WHEN RDC THEN BEX;JUMP
XD=(A-REG);MAR=A-ADDR;SAR=SHIFT AMT.

X
X ***** COMPARE SUBROUTINES *****
SETCOMPARE: XCOMPARE A(B) WITH A(A) AND SET CD BITS
HALFFETCH=AMPCR
IF LC1 THEN STEP ELSE SKIP XLC1=> CALLED BY OPR 44,46,47
IFETCH-1=AMPCR
ELSE OPR 74X4,6,7
SETCDRET: XENTER HERE AND RETURN WHERE CALLED FROM
A3 EOL B;SET LC2
IF TRUE THEN A2 OR B100 C=A2;CSAR;SKIP XSET EQUAL BIT
A2 AND B011 C=A2;CSAR XCLEAR EQUAL BIT
XGREATER THEN EQUAL TO BIT NOW IN MSB
31=5AR
A3 XOR D
A3 LSS B; IF MST THEN STEP ELSE SKIP
IF FALSE THEN A2 AND B011 C = A2;JUMP ELSE SKIP
IF TRUE THEN A2 AND B011 C=A2;JUMP
A2 OR B100 C=A2;JUMP
X42 RESTORED TO NORMAL BEFORE EXIT TO WHERE CALLED FROM

X
0708CC00 D
07090C00 D
*****07100C00 D
07110C00 D
07120C00 D
07130000 D
0714CC00 D
0715CC0C D
07160C00 D
0717CC00 D
07180C00 D
07190000 D
07200C00 D
*****07210C00 D
0722CC00 D
0723CC00 D
07240C00 D
07250C00 D
07260000 D
07270C00 D
0728CC00 D
07290C00 D
07300C00 D
0731CC00 D
0732CC00 D
07330C00 D
0734CC0C D
07350C00 D
0736CC00 D
07370C00 D
07380C00 D
07390C00 D
0740CC00 D
0741CC00 D
07420000 D
07430000 D
07440000 D
0745CC00 D
07460C00 D
0747CC00 D
07480C00 D
07490C00 D
*****07500C00 D
0751CC00 D
07520C00 D
07530C00 D
07540000 D
07550C00 D
07560C00 D
07570C00 D
07580C0C D
07590C00 D
0760CC00 D
07610C0C D
07620C00 D
07630C00 D
07640C00 D
07650C00 D
0766CC0C D
0767CC00 D

```



```

0293 4809 0000 1C31 00F0
0294 11FC 0000 0000 0080
0295 4809 0C41 8B02 00F0
0296 5C09 E0C0 1030 00F0
0297 8829 00C0 0030 00F0
0298 294C 0000 0030 0040

*****
COUNTONES: %COUNT NUMBER OF ONES IN B AND RETURN COUNT IN A3,MIR
O=A3,LC1R
31=L1111=SAR
CHECKLSB: B C=B,JNC
IF LST THEN A3+1=A3,MIR
IF COV THEN JUMP
CHECKLSB-1=MPCR

*****
MULTIPLY: %A2 AND B CONTAIN OPERATORS
O=A1,A3,LC1R
31=L1111=SAR
O EOL B,IF LC1
IF FALSE THEN A2 XOR B,STEP ELSE JUMP
A2,IF HST THEN SET LC1 %TEST FOR DIFFERENT SIGNS
IF HST THEN NOT A2=A2 %TEST FOR NEGATIVE
B %TEST FOR NEGATIVE
IF HST THEN NOT B=B
MULT1: A2 R=A2,INC
IF LST THEN A1+B R=A1,SKIP %A1=MOST SIGNIFICANT
A1 R=A1
IF LST THEN A3 OR B100=A3 %A3=LEAST SIGNIFICANT
IF NOT COV THEN A3 R=A3,STEP ELSE SKIP
MULT1-1=MPCR
IF LC1 THEN NOT A3=A3,STEP ELSE JUMP
NOT A1=A1,JUMP

*****
DIVIDE: %B=DIVISOR, A3=MOST DIVIDEND, MAR=LEAST DIVIDEND ADDRESS
O=A1,A2,LC1R,SET LC2
30=L11,COMP 1=SAR
B EOL O
A3, IF TRUE THEN SET LC1,JUMP %OVERFLOW=>DIVISION BY ZERO
IF HST THEN NOT A3=A3,SET LC1
B,MRI,IF RDC
IF HST THEN BIFF=E %DIVISOR NEGATIVE
DIV1: A3 L=A3
IF HST THEN A1 OR B001=A1
A1 LSS B011
IF FALSE THEN A2 OR B001=A2,STEP ELSE SKIP
A1-B011=A1
INC,IF NOT COV THEN SKIP
DIV2-I=MPCR
A1 L=A1
A2 L=A2
IF HST THEN SET LC1,JUMP
DIV1-I=MPCR
DIV2: IF LC2 THEN A2 L=A2,LC1R,SKIP
DIV3-I=MPCR
B=A3
WHEN RDC THEN A1 L=A1,BEX %B=LEAST DIVIDEND
A3 XOR B=A3 %SWAP B AND A3
A3 XOR B=B
A3 XOR B=A3
IF LC1 THEN NOT A3=A3,SET LC1
DIV1-I=MPCR

```



```

C2C4 4809 C040 0030 00F0
C2C5 4019 0000 0030 00F0
C2C6 00F0 0000 0030 0040
C2C7 2000 C0C2 2030 00F0
C2C8 4820 A0C2 4030 00F0
C2C9 2029 A002 4030 00F0
C2CA 4820 C002 2030 00F0

02CB 4809 C3C1 1000 00F0
C2CC A000 0000 0000 0020
C2CD 4809 E0C0 0030 00F0
C2CE 4808 0000 0000 00FC
C2CF 0000 0000 0000 0060

02D0 3809 2001 0800 00F0
02D1 3070 0003 0030 0090
02D2 48C9 C04E 2030 00F0
02D3 0AA0 0000 0030 0040

02D4 4809 00C0 0038 00F0
02D5 00F0 0000 0030 00E0
02D6 A809 00C0 0000 0080
02D7 AC08 00C0 0030 00F0
02D8 4809 0C42 0030 00F0
02D9 6400 0C40 1000 00F0
C2DA 4809 E0DE 0030 00F0
02DB 2300 0030 0030 0060
02DC 4809 0000 0038 00F0
02DD 018C 00C0 0030 00E0
02DE A809 0000 0030 0080
02DF AC08 00C0 0030 00FC
02E0 4809 0C40 0030 00F0
02E1 8070 0003 0000 00A0
02E2 4809 A156 4030 00F0
02E3 0010 0000 0030 00C0
02E4 3829 00C0 0030 00FC
02E5 0020 0000 0030 0040
02E6 4809 00C0 0038 00F0
02E7 018C 00C0 0030 00E0
02E8 A809 0000 0000 0080
02E9 AC08 00C0 0030 00FC
02EA 4809 0C41 0038 00F0
02EB 0190 0000 0000 00A0
02EC A809 0C40 0030 00FC
02ED 4C19 0018 0000 00F0
02EE 4809 0000 0030 00F0
02EF AC08 AC53 8080 00F0
02F0 4809 0C40 4000 00FC
02F1 4809 A000 9000 00F0
02F2 1070 00C0 0030 00A0
C2F3 4809 E156 1030 00F0
02F4 4809 E140 0030 00F0
02F5 0080 0000 0030 00E0
02F6 A809 0000 0000 0080
02F7 AC08 0C40 1000 00F0
02F8 4809 EC40 0030 00F0
02F9 2300 00C0 0030 0060

DIV3: B IF NOT MST THEN SKIP
DIVNEG-1=MPCR
IF LC1 THEN NOT A2=A2;STEP ELSE JUMP
NOT A1=A1;JUMP
DIVNEG: IF LC1 THEN NOT A1=A1;JUMP
NOT A2=A2;JUMP

%
%
% *****ROUTINES FOR REPEAT AND INDIRECTION CONTROL *****
GC2CHECK1: A2 L = A3
COMP 10=SAR
A3 %CHECK BIT 21=REPEAT BIT
IF MST THEN STEP ELSE SKIP % CHECK FOR REPEAT MODE
RPTCHECK-1 = CPCR

GC2CHECK1: LIT L = B; IF LC2
7 = LIT; COMP 21 = SAR
A2 NIM B = A2; RESET GC2 % CLEAR A2(23-21)
IFETCH1- 1 = NPCR

RPTCHECK: LMAR % B(7)
15 = LIT
MR1; IF RDC
WHEN RDC THEN BEX
NOT B
IF NOT ABT THEN B=A3;STEP ELSE JUMP %(B(7))=C;RETURN GC2CHECK
A3-1 = MIR % DEC B(7)
OUTPUT1-1 = CPCR
LMAR %REPEAT INSTR.
MR1; IF RDC % REPEAT INSTR.
WHEN RDC THEN BEX
B R=A1;MIR
7 = LIT; 23 = SAR
A1 AND LIT = A1 % A-FIELD
ACMP0-1 = AMPCR % CHECK A-FIELD
IF LC2 THEN JUMP % FOR TERMINATE
ANCHP0-1 = MPCR % CONDITIONS
RPTCHECK1: LMAR %CURRENT INSTR.
RINST=LIT
MR1;IF RDC
WHEN RDC THEN BEX
B L = B;LMAR
CINST=LIT;16=SAR
B;MR1;IF RDC
IF MST THEN B11 R=A1;SKIP
O=A1
WHEN RDC THEN A1 OR B=C;MIR;BEX
B = A1 % RESTORE C.I.
A1 R = A3
7 = LIT; 17 = SAR
A3 AND LIT = A3;BMI % E-FIELD
A3 + LIT = MAR
BRASE = LIT
MR1; IF RDC
WHEN RDC THEN B = A3;BEX
A3 + B = MIR
OUTPUT1-1 = CPCR % 6(B)+SY=>B(B)

```


| | | | | |
|--------|------|------|------|------|
| 4002FA | 4809 | 0000 | 0038 | 00FA |
| 4002FB | 0180 | 0000 | 0030 | 00E0 |
| 4002FC | 4809 | 0000 | 0000 | 0080 |
| 4002FD | 4809 | 0000 | 0030 | 00F0 |
| 4002FE | 4809 | EC40 | 0030 | 00F0 |
| 4002FF | 2300 | 0000 | 0030 | 0060 |
| 400300 | 0830 | 0000 | 0000 | 0040 |
| 400301 | 4809 | 0640 | 0030 | 00FC |
| 400302 | 4809 | A000 | 9030 | 00F0 |
| 400303 | 1070 | 0000 | 0000 | 00A0 |
| 400304 | 4809 | E156 | 1030 | 00F0 |
| 400305 | 4809 | E140 | 000C | 00F0 |
| 400306 | 0080 | 00C0 | 0030 | 00E0 |
| 400307 | 4809 | A0C1 | 0800 | 48F0 |
| 400308 | 9160 | 0000 | 0030 | 0090 |
| 400309 | AC08 | 0C40 | 9030 | 00F0 |
| 40030A | 4809 | EC40 | 1008 | 00F0 |
| 40030B | 4809 | 0000 | 0030 | 0080 |
| 40030C | 0180 | 0000 | 0030 | 00E0 |
| 40030D | AC08 | 0000 | 000C | 00F0 |
| 40030E | 4809 | EC40 | 0038 | 00F0 |
| 40030F | 2300 | 0000 | 0030 | 0060 |
| 400310 | 4809 | 0C40 | 0040 | 00F0 |
| 400311 | 4809 | 0000 | 0000 | 00F0 |
| 400312 | 4820 | 0000 | 0030 | 00F0 |
| 400313 | 4809 | A000 | 9000 | 00F0 |
| 400314 | 1070 | 0000 | 0000 | 00A0 |
| 400315 | 4809 | E156 | 1030 | 00F0 |
| 400316 | 4809 | E140 | 000C | 00F0 |
| 400317 | 0080 | 0000 | 0000 | 00EC |
| 400318 | 4809 | A0C1 | 0800 | 48F0 |
| 400319 | 91FC | 0000 | 0000 | 0090 |
| 40031A | AC08 | 0C40 | 9000 | 00F0 |
| 40031B | 4809 | EC40 | 1000 | 00F0 |
| 40031C | 4809 | A000 | 8800 | 00FC |
| 40031D | 9070 | 00C0 | 0030 | 0090 |
| 40031E | 4809 | 2C56 | 0830 | 00F0 |
| 40031F | 4809 | 2C40 | 000C | 00F0 |
| 400320 | 0100 | 0000 | 0000 | 00E0 |
| 400321 | 4809 | 0000 | 0000 | 0080 |
| 400322 | AC08 | 0000 | 0000 | 00F0 |
| 400323 | 4820 | EC40 | 101C | 00F0 |
| 400324 | 4809 | A000 | 9030 | 00F0 |
| 400325 | 1070 | 0000 | 0030 | 00A0 |
| 400326 | 4809 | E156 | 1030 | 00F0 |
| 400327 | 4809 | E140 | 000C | 00F0 |
| 400328 | 0100 | 0000 | 0000 | 00A0 |
| 400329 | 4809 | A0C1 | 1030 | 0080 |
| 40032A | AC08 | E000 | 9030 | 00F0 |
| 40032B | 4820 | EC40 | 101C | 00F0 |
| 40032C | 4809 | A000 | 9030 | 00F0 |
| 40032D | 1070 | 0000 | 0000 | 00A0 |
| 40032E | 4809 | E156 | 1030 | 00F0 |
| 40032F | 4809 | E140 | 000C | 00F0 |
| 400330 | 0080 | 0000 | 0000 | 00A0 |
| 400331 | 4809 | A001 | 0800 | 08F0 |

| | | | | | | | | |
|-----------------------------|------|------|------|------|---------------------------------|------------------|----------|---|
| 0332 | AC08 | 0C40 | 8C80 | 00F0 | WHEN RDC THEN B R = MIR,BEX | % SY => MIR | 09480C00 | D |
| 0333 | 4809 | E140 | 003C | 00F0 | A3 + LIT = HAR | % S(B) | 09490C00 | D |
| 0334 | 0100 | 0000 | 0000 | 00E0 | SBASE = LIT | | 09500C00 | D |
| 0335 | 4809 | 0C40 | 1000 | 08F0 | B = A3,BMI; MR1; IF RDC | % B(B)>A3,SY=>B | 09510C00 | D |
| 0336 | AC08 | EC40 | 1C00 | 00F0 | WHEN RDC THEN A3 + B = A3,BEX | | 09520C00 | D |
| 0337 | 482D | EC43 | 101C | 00F0 | A3 + B = A3,MAR2; JUMP | % SY+B(B)+S(B) | 09530C00 | D |
| INDIRECT: | | | | | | | 09540C00 | D |
| 0338 | 4809 | A000 | 9C00 | 00F0 | A1 R = A3 | % OPCODE | 09550C00 | D |
| 0339 | 2300 | 0000 | 0030 | 00B0 | 48-LIT;26=SAR | | 09560000 | D |
| 033A | 4809 | E15E | 0000 | 00F0 | A3 LSS LIT | | 09570C00 | D |
| 033B | 7019 | 0000 | 0030 | 00F0 | IF TRUE THEN SKIP | | 09580C00 | D |
| 033C | 0B3C | 0003 | 0000 | C040 | IFETCH2-1 = MPCR | % HALF-WORD INST | 09590C00 | D |
| INDIRECT1: | | | | | | | 09600C00 | D |
| FEINORM-1 = CPCR | | | | | % Y=Y+P(B)+S(S) | | 09610C00 | D |
| INDIRECT2: | | | | | | | 09620C00 | D |
| A1 R = A1,CSAR; MR2; IF RDC | | | | | % Y => MAR2 | | 09630C00 | D |
| 20 = SAR | | | | | | | 09640C00 | D |
| WHEN RDC THEN A1 L = A1,BEX | | | | | | | 09650C00 | D |
| B = MIR, CSAR | | | | | % (Y) = MIR | | 09660C00 | D |
| B L = B,CSAR | | | | | | | 09670C00 | D |
| B R = B | | | | | | | 09680C00 | D |
| A1 OR B = A1 | | | | | % REPLACE A1(19-0) | | 09690C00 | D |
| A1 R = A3,BMI | | | | | % (Y)=E | | 09700C00 | D |
| 16 = SAR | | | | | | | 09710C00 | D |
| A3 | | | | | | | 09720000 | D |
| IF LST THEN SKIP | | | | | % A1(16)=1? | | 09730C00 | D |
| INDIRECT3-1 = MPCR | | | | | % A1(16)=0 | | 09740C00 | D |
| B R = A3 | | | | | % A1(16)=1 | | 09750C00 | D |
| ONE = LIT; 29 = SAR | | | | | | | 09760C00 | D |
| A3 GTR LIT | | | | | | | 09770C00 | D |
| IF FALSE THEN SKIP | | | | | | | 09780C00 | D |
| INDIRECT1-1 = MPCR | | | | | | | 09790000 | D |
| FETCHOPT1-1 = AMPCR | | | | | | | 09800C00 | D |
| A3 | | | | | | | 09810C00 | D |
| IF LST THEN CALL ELSE SKIP | | | | | | | 09820C00 | D |
| SKIP | | | | | | | 09830C00 | D |
| FETCHOPT0-1 = CPCR | | | | | | | 09840C00 | D |
| INDIRECT2-1 = MPCR | | | | | | | 09850C00 | D |
| INDIRECT3: | | | | | % Y => MAR2,(Y)=>B,MIR | | 09860C00 | D |
| LMAR | | | | | | | 09870C00 | D |
| IAW=LIT | | | | | | | 09880C00 | D |
| OUTPUT1-1 = CPCR | | | | | % SAVE (Y) = ICW | | 09890C00 | D |
| LMAR | | | | | | | 09900C00 | D |
| IAWADDR=LIT | | | | | | | 09910C00 | D |
| OUTPUT1-1 = CPCR | | | | | % SAVE Y = ADDR ICW | | 09920C00 | D |
| IF GC2 THEN SKIP | | | | | % REPEAT | | 09930C00 | D |
| INDIRECT4-1 = MPCR | | | | | | | 09940C00 | D |
| S6YFEICH-1 = CPCR | | | | | % Y = Y+B(P)+S(6) | | 09950C00 | D |
| A1 = MIR,LMAR | | | | | | | 09960C00 | D |
| CINSTR=LIT | | | | | | | 09970C00 | D |
| OUTPUT1-1 = CPCR | | | | | % SAVE C.I. | | 09980C00 | D |
| IFETCH2-1 = MPCR | | | | | | | 09990000 | D |
| INDIRECT4: | | | | | | | 10000C00 | D |
| B R = A3 | | | | | % (Y)31-29 | | 10010C00 | D |
| ONE = LIT; 29 = SAR | | | | | | | 10020C00 | D |
| A3 GTR LIT | | | | | | | 10030C00 | D |
| IF FALSE THEN SKIP | | | | | % SKIP FOR OPTIONAL INDIRECTION | | 10040C00 | D |
| INDIRECT6-1 = MPCR | | | | | % NORMAL OR CA | | 10050C00 | D |
| B100 R = B | | | | | | | 10060C00 | D |
| ZERO=LIT; 8=SAR | | | | | | | 1007000C | D |
| 033E | 4809 | A0C3 | C000 | 4CF0 | | | | |
| 033F | 8000 | C000 | 0030 | 0020 | | | | |
| 0340 | AC08 | A001 | 4C30 | C0F0 | | | | |
| 0341 | 4809 | 0C40 | 0030 | 40F0 | | | | |
| 0342 | 4809 | 0C41 | 0B30 | 40FC | | | | |
| 0343 | 4809 | 0C40 | 8B30 | C0F0 | | | | |
| 0344 | 4809 | AC5C | 4C30 | 00F0 | | | | |
| 0345 | 4809 | A000 | 9000 | 00F0 | | | | |
| 0346 | 0000 | 0000 | 0030 | 0020 | | | | |
| 0347 | 4809 | E000 | 0030 | C0F0 | | | | |
| 0348 | 5819 | 0000 | 0000 | C0FC | | | | |
| 0349 | 0D3C | 0000 | 0030 | C040 | | | | |
| 034A | 4809 | 0C40 | 9030 | C0F0 | | | | |
| 034B | 9010 | 0000 | 0030 | 00B0 | | | | |
| 034C | 4809 | E159 | 0030 | C0F0 | | | | |
| 034D | 7019 | 0000 | 0030 | 00F0 | | | | |
| 034E | 33C0 | 0000 | 0030 | 0040 | | | | |
| 034F | 32B0 | 0000 | 0000 | 00C0 | | | | |
| 0350 | 4809 | E000 | 0030 | C0F0 | | | | |
| 0351 | 5833 | 0000 | 0000 | 00F0 | | | | |
| 0352 | 4818 | 00C0 | 0000 | C0F0 | | | | |
| 0353 | 3230 | 0003 | 0000 | C060 | | | | |
| 0354 | 3300 | 0000 | 0000 | 0040 | | | | |
| 0355 | 4809 | 00C0 | 0C38 | 00FC | | | | |
| 0356 | 01A0 | 0000 | 0030 | 00E0 | | | | |
| 0357 | 2300 | 0000 | 0030 | C060 | | | | |
| 0358 | 4809 | 00C0 | 0008 | C0F0 | | | | |
| 0359 | 01C0 | 0000 | 0000 | 00E0 | | | | |
| 035A | 230C | 0000 | 0030 | 0060 | | | | |
| 035B | 1819 | 0000 | 0000 | 00F0 | | | | |
| 035C | 0D40 | 0000 | 0030 | C040 | | | | |
| 035D | 3000 | 0000 | 0030 | 0060 | | | | |
| 035E | 4809 | A000 | 0038 | 00F0 | | | | |
| 035F | 019C | 0000 | 0030 | C0E0 | | | | |
| 0360 | 2300 | 0000 | 0030 | 0060 | | | | |
| 0361 | 0B3C | 0000 | 0030 | 0040 | | | | |
| 0362 | 4809 | 0C40 | 9030 | 00F0 | | | | |
| 0363 | 9010 | 0000 | 0000 | 00B0 | | | | |
| 0364 | 4809 | E159 | 0C0C | C0F0 | | | | |
| 0365 | 7019 | 0000 | 0000 | 00F0 | | | | |
| 0366 | 0D3C | 0000 | 0030 | 0040 | | | | |
| 0367 | 4809 | 1800 | 8B00 | 00F0 | | | | |
| 0368 | 0000 | 00C0 | 0C00 | 0090 | | | | |


```

0369 4809 CC5C 2000 00F0
036A 3230 0000 0030 00C0
036B 4809 E152 0C00 00F0
036C 6833 0000 0000 00F0
036D 481B 0000 0000 00F0
036E 3280 0000 0000 00C0
036F 4889 E0C0 0098 00F0
0370 0180 0000 0030 00E0
0371 2300 0000 0030 0060
0372 0830 0000 0030 0040
0373 4809 E000 9000 00F0
0374 1000 0000 0000 0000
0375 4809 E000 0000 00F0
0376 580E 00C0 0030 00F0
0377 0D60 0000 0030 0040
0378 4809 18C0 8800 00F0
0379 1000 0000 0000 0010
037A 4809 CC5C 2000 00F0
037B 3120 00C0 0000 0060
037C 4809 E000 0090 00F0
037D 36E0 0000 0030 0040
037E 4809 A0C0 903C 00F0
037F 2000 0000 0030 003C
0380 0D70 0000 0000 0060
0381 3819 0000 0030 00F0
0382 0A5C 0000 003C 0040
0383 4809 2001 0800 00F0
0384 2030 00C0 0000 0090
0385 4809 CC5C 2000 00F0
0386 37AC 0000 0000 0040
0387 2920 0000 0030 0060
0388 4809 A640 0090 00F0
0389 0D8C 0000 003C 00C0
038A 4809 A0C0 003C 00F0
038B A824 0000 0030 00F0
038C AC08 0000 00C0 00F0
038D 4836 0C40 0090 00F0
038E 2CFC 00C0 0000 0040
038F 2E50 0000 0030 0040
0390
0394
0398 4809 0C52 0000 00F0
0399 602F 0000 0000 00F0
039A 4809 0C52 0000 00F0
039B 682F 0000 0000 00F0
039C 4839 0C52 000C 00F0
039D 682F 0000 0030 00F0
039E 4829 0C53 0000 00F0
039F 782F 0000 0030 00F0
03A0 483F 0000 0000 00F0
03A1 4809 E000 0000 00F0
03A2 502F 0000 0030 00F0

```

```

A2 OR B=A2
FETCHOPT0-1 = AMPCR
A3 EOL LIT
IF TRUE THEN CALL ELSE SKIP
SKIP
FEICHOPI1-1 = CPCR
INDIRECT15:
A3 = MIR,LMAR, SET GC2
YADDR=LIT
OUTPUT1-1 = CPCR
IFETCH2-1 = MPCR
INDIRECT16:
A3 R = A3
1 = SAR
A3
IF LST THEN STEP ELSE SKIP
INDIRECT18-1 = MPCR
B130 R = B
9=SAR
A2 OR B=A2
INDIRECT17:
FETNORM-1 = CPCR
A3 = MIR
INDIRECT15-1 = MPCR
INDIRECT18:
A1 R = A3
26 = SAR
CONDHECK-1 = CPCR
IF LC2 THEN SKIP
IFETCH-1 = MPCR
LIT L=B
3=LIT, COMP 22=SAR
A2 OR B=A2
INDIRECT1-1 = MPCR
ANCHP0:
COUNTONES-1=CPCR
A1+AMPCR=AMPCR
AOP-1=AMPCR
A1=MAR
MR1EXEC,IF RDC
WHEN RDC THEN BEX
B,CALL
GC2CHECK1-1=MPCR
RPTCHECK1-1=MPCR
S-CODE
AOP: AOP0-1=AMPCR. AOP1-1=AMPCR. AOP2-1=AMPCR. AOP3-1=AMPCR
AOP4-1=AMPCR. AOP5-1=AMPCR. AOP6-1=AMPCR. AOP7-1=AMPCR
$POP CODE
AOP0: 0 EOL B
IF FALSE THEN JUMP ELSE RETN
AOP1: 0 EOL B
IF TRUE THEN JUMP ELSE RETN
AOP2: 0 EOL B,IF MST THEN RETN
IF TRUE THEN JUMP ELSE RETN
AOP3: 0 GTR B,IF MST THEN JUMP
IF TRUE THEN JUMP ELSE RETN
AOP4: RETN
AOP5: A3. IF NOT LST THEN JUMP ELSE RETN
XSET A2(23)
XSAVE Y AT 19
X A3=(Y)31-29
X A3 = (Y)31,3C
X NORMAL INDIRECTION
XSET A2(22)
X CHECK CA
X OPCODE
XSET BITS 22,23
XSET A2(22,23)
XA1=A-REG REFERENCED
XA3=ONES COUNT
XADDRESS OF A-REG
XREAD A-REG,EXECUTE CODE HANDLER
XB=(A)
XEXECUTE CODE HANDLER
XTERMINATE
XNO TERMINATE

```


| | | | | | | | | | |
|------|------|------|------|------|--|--|----------------------------|----------|---|
| 03A3 | 4809 | E000 | 0030 | 00F0 | A0P6: | A3. | IF LST THEN JUMP ELSE RETN | 10670000 | 0 |
| 03A4 | 582F | 0000 | 0030 | 00F0 | A0P7: | RETN | | 10680000 | 0 |
| 03A5 | 483F | 0000 | 0000 | 00F0 | * | | | 10690000 | 0 |
| 03A6 | 4809 | A640 | 0040 | 00F0 | ACMP0: | %CHECK CONDITION BITS FOR TERMINATING REPEAT | | 10700000 | 0 |
| 03A7 | 0E10 | 0003 | 0030 | 00C0 | A1+AMPCR=AMPCR | %A1=A-REG REFERENCED | | 10710000 | 0 |
| 03A8 | 3000 | 0000 | 0000 | 0030 | AC0-1=AMPCR | | | 10720000 | 0 |
| 03A9 | 4809 | C001 | 7030 | 00F0 | COMP 1=5AR | | | 10730000 | 0 |
| 03AA | 4824 | 0003 | 0030 | 00F0 | A2 C=A3 | %EOL IS LSB/GE0 IS MSB/LIMITS 2ND MSB | | 10740000 | 0 |
| 03AB | 4836 | E000 | 0030 | 00F0 | EXEC | | | 10750000 | 0 |
| 03AC | 2CF0 | 0000 | 0030 | 0040 | A3/CALL | %CALL CONDITION HANDLER | | 10760000 | 0 |
| 03AD | 2E50 | 0000 | 0030 | 0040 | GC2CHECK1-1=AMPCR | %TERMINATE | | 10770000 | 0 |
| 03AE | | | | | RPTCHECK1-1=HPCR | | | 10780000 | 0 |
| 03B2 | | | | | \$-CODE | | | 10790000 | 0 |
| 03B6 | 502F | 0000 | 0030 | 00F0 | AC00-1=AMPCR. AC01-1=AMPCR. AC02-1=AMPCR. AC03-1=AMPCR | | | 10800000 | 0 |
| 03B7 | 582F | 0000 | 0030 | 00F0 | AC04-1=AMPCR. AC05-1=AMPCR. AC06-1=AMPCR. AC07-1=AMPCR | | | 10810000 | 0 |
| 03B8 | 540F | E000 | 0030 | 00F0 | \$POP CODE | | | 10820000 | 0 |
| 03B9 | 482F | 0000 | 0000 | 00F0 | AC00: | IF NOT LST THEN JUMP ELSE RETN | | 10830000 | 0 |
| 03BA | 482F | 0003 | 0000 | 00F0 | AC01: | IF LST THEN JUMP ELSE RETN | | 10840000 | 0 |
| 03BB | 402F | 0000 | 0000 | 00F0 | AC02: | IF NOT LST THEN A3/STEP ELSE RETN | | 10850000 | 0 |
| 03BC | 540F | E000 | 0030 | 00F0 | AC03: | IF HST THEN JUMP ELSE RETN | | 10860000 | 0 |
| 03BD | 402F | 0000 | 0030 | 00F0 | AC04: | IF NOT HST THEN JUMP ELSE RETN | | 10870000 | 0 |
| 03BE | 48C9 | E00A | 0030 | 00F0 | AC05: | IF NOT LST THEN A3/STEP ELSE RETN | | 10880000 | 0 |
| 03BF | 482F | 0000 | 0030 | 00F0 | AC06: | IF NOT HST THEN JUMP ELSE RETN | | 10890000 | 0 |
| 03C0 | 4809 | E00A | 0000 | 00F0 | AC07: | IF HST THEN JUMP ELSE RETN | | 10900000 | 0 |
| 03C1 | 402F | 0003 | 0030 | 00F0 | * | A3 0A0 0 | | 10910000 | 0 |
| 03C2 | 4809 | 0641 | 0010 | 24F0 | CONOCHECK: | %TEST LIMITS BIT 2ND MSB | | 10920000 | 0 |
| 03C3 | 0000 | 0000 | 0000 | 0030 | AMPCR L=BR2/ASE | %TEST LIMITS BIT 2ND MS0 | | 10930000 | 0 |
| 03C4 | 4809 | E640 | 0040 | 00F0 | COMP 8=5AR | | | 10940000 | 0 |
| 03C5 | 0EAC | 0000 | 0000 | 00C0 | A3+AMPCR=AMPCR | %RETURN LC1=>REPEATABLE | | 10950000 | 0 |
| 03C6 | 2809 | 0C41 | 8830 | 00F0 | CONOTABLE-1=AMPCR | %LC2 => CHAR.ADRRESSABLE | | 10960000 | 0 |
| 03C7 | 8000 | 00C0 | 0030 | 0020 | B C=8/JIF LC1 | %SAVE RETURN ADDRESS IN BR2 | | 10970000 | 0 |
| 03C8 | 3824 | 0000 | 0030 | 00F0 | 20=5AR | %RETURN SAVED IN BR2 | | 10980000 | 0 |
| 03C9 | 4809 | 2640 | 0040 | 00F0 | SETUPCD: EXEC/JIF LC2 | %SET CONDITION CODE | | 11000000 | 0 |
| 03CA | 0E80 | 0000 | 0000 | 00C0 | LIT+AMPCR=AMPCR | | | 11010000 | 0 |
| 03CB | 4809 | 2C48 | 0035 | 00F0 | SETCON0-1=AMPCR | %SETUP TO EXECUTE CONDITIONS | | 11020000 | 0 |
| 03CC | 0070 | 0000 | 0030 | 0090 | LIT NAN B=CTR | | | 11030000 | 0 |
| 03CE | 0EC0 | 0000 | 0000 | 00C0 | 7=LIT/8=5AR | %SAVE SUBOPCODE IN CTR | | 11040000 | 0 |
| 03CF | 49C9 | 0000 | 0000 | 00F0 | JUMP | %EXECUTE SETCOND + VALUE IN TABLE | | 11050000 | 0 |
| 03D0 | 4820 | 0000 | 0030 | 00F0 | SETCOND: SPECCON0-1=AMPCR | %X=LIT=> 0? OR 03 C0PCODE | | 11060000 | 0 |
| 03D1 | 49C9 | 0000 | 0000 | 00F0 | SET LC1 | %X=LIT R OR CA | | 11070000 | 0 |
| 03D2 | 4809 | 0F40 | 8040 | 00F0 | SET LC2/SKIP | %2=LIT NOR P BUT CA | | 11080000 | 0 |
| 03D3 | 4809 | 0000 | 0030 | 00F0 | SET LC1 | %3=LIT NOT CA BUT R | | 11090000 | 0 |
| 03D4 | 4820 | 0000 | 0030 | 00F0 | CONDEXIT: BHAR R=AMPCR | %4=LIT NOT CA & NOT R | | 11100000 | 0 |
| 03D5 | | | | | BNI | %RESET B | | 11110000 | 0 |
| 03D6 | | | | | JUMP | %RETURN TO CALLING ROUTINE | | 11120000 | 0 |
| 03D7 | 4809 | 6640 | 0040 | 00F0 | SPECCON0: | | | 11130000 | 0 |
| 03D8 | 0E00 | 0000 | 0030 | 00C0 | CTR+AMPCR=AMPCR | %MUST BE 02/03 | | 11140000 | 0 |
| 03D9 | 4809 | E152 | 0030 | 00F0 | SUBTABLE-1=AMPCR | | | 11150000 | 0 |
| 03DA | 0030 | 0000 | 0030 | 00E0 | A3 EQL LIT | %CHECK FOR 03 | | 11160000 | 0 |
| 03DB | 6C08 | 2640 | 0040 | 00F0 | 3=LIT | | | 11170000 | 0 |
| 03DC | 0080 | 00C0 | 0030 | 00EC | IF TRUE THEN LIT+AMPCR=AMPCR/STEP ELSE SKIP | | | 11180000 | 0 |
| | 4809 | 0000 | 0030 | 00F0 | 8=LIT | %OFFSET PAST 02 CODES | | 11190000 | 0 |
| | | | | | STEP | | | 11200000 | 0 |
| | | | | | SETUPCD-1=HPCR | %USE CODE FOR REGULAR OPCODES | | 11210000 | 0 |
| | | | | | \$-CODE | | | 11220000 | 0 |
| | | | | | | | | 11230000 | 0 |
| | | | | | | | | 11240000 | 0 |
| | | | | | | | | 11250000 | 0 |

| | | | | | | | | | | | |
|------|------|------|------|-------|-------------------------------|---|-------|-----------------------------------|-------|----------|---|
| 0452 | 4820 | 0C40 | 0030 | 00F0 | B=MIR;JUMP | % | ***** | FLOATING POINT SUBROUTINES | ***** | 11860C00 | D |
| 0453 | 224C | 0000 | 0030 | 0040 | UNWRITTEN-1=MPCR | % | ***** | %PRINT ERROR AND RETURN TO IFETCH | ***** | 11870C00 | D |
| 0454 | 127C | 0000 | 0030 | 006C | Y2FETCH-1=CPCR | % | ***** | EXTERNAL FUNCTION | ***** | 11880C00 | D |
| 0455 | 4809 | 0F40 | 0830 | 00F0 | BHAR=B | % | ***** | %B=ADDRESS OF Y+(B)+(S) | ***** | 11890C00 | D |
| 0456 | 246C | 0000 | 0030 | 0060 | GETF-1=CPCR | % | ***** | %RETURNS F2-FIELD IN A3 | ***** | 11900C00 | D |
| 0457 | 48C9 | E640 | 0090 | 00F0 | A3+AMPCR=AMPCR | % | ***** | %SETUP FOR JUM | ***** | 11910C00 | D |
| 0458 | 0600 | 0000 | 0030 | 00C0 | OPR07XX-1=AMPCR | % | ***** | %ISOLATE A FIELD | ***** | 11920C00 | D |
| 0459 | 4809 | AC00 | 9000 | 00F0 | A1 R=A3 | % | ***** | | ***** | 11930C00 | D |
| 045A | 8070 | 0000 | 0030 | 00A0 | 7=LIT;23=SAR | % | ***** | | ***** | 11940C00 | D |
| 045B | 4824 | E156 | 1030 | 00F0 | A3 AND LIT=A3;EXEC | % | ***** | | ***** | 11950C00 | D |
| 045C | 4820 | 0000 | 0030 | 00F0 | JUMP | % | ***** | | ***** | 11960C00 | D |
| 045D | 0CF0 | 0000 | 0030 | 0060 | ***** | % | ***** | LOAD A | ***** | 12000C00 | D |
| 045E | 24C0 | 00C0 | 0030 | 0060 | YFETCH-1 = CPCR | % | ***** | % LA | ***** | 12010C00 | D |
| 045F | 22F0 | 0000 | 0030 | 004C | GETA3-1=CPCR | % | ***** | %PUTS PROPER INFO INTO MIR | ***** | 12020C00 | D |
| 0460 | 0CF0 | 0000 | 0030 | 0060 | OUTPUT-1=MPCR | % | ***** | %RETURNS A-FIELD IN A3&MAR | ***** | 12030C00 | D |
| 0461 | 24C0 | 0000 | 0030 | 0060 | ***** | % | ***** | LOAD A AND INDEX B | ***** | 12040C00 | D |
| 0462 | 9809 | A000 | 9030 | 18F0 | YFETCH-1 = CPCR | % | ***** | % LXB | ***** | 12050C00 | D |
| 0463 | 1070 | 0000 | 0030 | 00A0 | GETA3-1=CPCR | % | ***** | % PUTS INFO INTO MIR | ***** | 12060C00 | D |
| 0464 | 4809 | E156 | 1000 | 00F0 | A1 R = A3+HW1; IF SAI | % | ***** | %RETURNS A-FIELD IN A3&MAR | ***** | 12070C00 | D |
| 0465 | 9C08 | E140 | 0030 | 00F0 | 17 = SAR; 7 = LIT | % | ***** | % WRITE A-REG | ***** | 12080C00 | D |
| 0466 | 0080 | 0000 | 0030 | 00A0 | A3 AND LIT = A3 | % | ***** | % P-FIELD = A3 | ***** | 12090C00 | D |
| 0467 | A809 | 0000 | 0030 | 008F0 | WHEN SAI THEN A3+LIT=MAR | % | ***** | | ***** | 12100C00 | C |
| 0468 | AC08 | 0000 | 0030 | 00F0 | BBASE = LIT; 16 = SAR | % | ***** | | ***** | 12110C00 | C |
| 0469 | 018C | 0000 | 0030 | 00E0 | MR1; IF RDC | % | ***** | | ***** | 12120C00 | D |
| 046A | 4809 | 0C46 | 0090 | 00F0 | WHEN RDC THEN BEX | % | ***** | | ***** | 12130C00 | D |
| 046B | 22F0 | 0000 | 0030 | 0040 | YADDR=LIT | % | ***** | | ***** | 12140C00 | D |
| 046C | 0CF0 | 0000 | 0030 | 0060 | 0 + B + 1 = MIR | % | ***** | | ***** | 12150C00 | D |
| 046D | 24C0 | 00C0 | 0000 | 0060 | OUTPUT-1=MPCR | % | ***** | LOAD DIFFERENCE | ***** | 12160C00 | D |
| 046E | A809 | 0000 | 0030 | 008F0 | ***** | % | ***** | % LDI | ***** | 12170C00 | D |
| 046F | AC08 | E0C0 | 0C30 | 00F0 | YFETCH-1 = CPCR | % | ***** | %RETURNS A-FIELD IN A3&MAR | ***** | 12180C00 | D |
| 0470 | 4809 | 0C40 | 1030 | 00F0 | GETA3-1=CPCR | % | ***** | | ***** | 12190C00 | D |
| 0471 | 4809 | E0C0 | 0C30 | 00F0 | MR1; IF RDC | % | ***** | | ***** | 12200C00 | D |
| 0472 | 4809 | 0C40 | 1030 | 00F0 | WHEN RDC THEN A3 +1 = MAR,BEX | % | ***** | % MAR = A+1 REG | ***** | 12210C00 | D |
| 0473 | 4809 | EC5E | 1090 | 00F0 | B = A3,BMI | % | ***** | | ***** | 12220C00 | D |
| 0474 | 7409 | E0DE | 1050 | 00F0 | A3 = MIR | % | ***** | | ***** | 12230C00 | D |
| 0475 | 2019 | 0000 | 0030 | 00F0 | B = A3,BMI | % | ***** | | ***** | 12240C00 | D |
| 0476 | 0EF0 | 0000 | 0030 | 0040 | A3 - B = A3,MIR | % | ***** | | ***** | 12250C00 | D |
| 0477 | 22F0 | 0000 | 0030 | 0040 | IF NOT ADV THEN A3-1=A3,MIR | % | ***** | | ***** | 12260C00 | D |
| 0478 | 0CF0 | 0000 | 0030 | 006C | IF NOT LC1 THEN SKIP | % | ***** | | ***** | 12270C00 | D |
| 0479 | 24C0 | 0000 | 0000 | 0060 | OPR36X1 -1 = MPCR | % | ***** | SUBTRACT A | ***** | 12280C00 | D |
| 047A | A809 | 0000 | 0030 | 008F0 | OUTPUT-1=MPCR | % | ***** | % ANA | ***** | 12290C00 | D |
| 047B | AC08 | 0000 | 0030 | 00F0 | ***** | % | ***** | %RETURNS A-FIELD IN A3&MAR | ***** | 12300C00 | D |
| 047C | 4809 | 0C40 | 1030 | 00F0 | YFETCH-1 = CPCR | % | ***** | | ***** | 12310C00 | D |
| 047D | 4809 | EC5E | 1090 | 00F0 | GETA3-1=CPCR | % | ***** | | ***** | 12320C00 | D |
| 047E | 7409 | E0DE | 1090 | 00F0 | MR1; IF RDC | % | ***** | | ***** | 12330C00 | D |
| 047F | 22F0 | 00C0 | 0030 | 0040 | WHEN RDC THEN BEX | % | ***** | | ***** | 12340C00 | D |
| | | | | | B = A3,BMI | % | ***** | | ***** | 12350C00 | D |
| | | | | | A3 - B = A3,MIR | % | ***** | | ***** | 12360000 | D |
| | | | | | IF NOT ADV THEN A3-1=A3,MIR | % | ***** | | ***** | 12370C00 | D |
| | | | | | IF NOT LC1 THEN SKIP | % | ***** | | ***** | 12380C00 | D |
| | | | | | OPR36X1 -1 = MPCR | % | ***** | | ***** | 12390C00 | D |
| | | | | | OUTPUT-1=MPCR | % | ***** | | ***** | 12400000 | D |
| | | | | | ***** | % | ***** | | ***** | 12410C00 | D |
| | | | | | YFETCH-1 = CPCR | % | ***** | | ***** | 12420C00 | D |
| | | | | | GETA3-1=CPCR | % | ***** | | ***** | 12430C00 | D |
| | | | | | MR1; IF RDC | % | ***** | | ***** | 12440C00 | D |
| | | | | | WHEN RDC THEN BEX | % | ***** | | ***** | 12450000 | D |
| | | | | | B = A3,BMI | % | ***** | | ***** | | |
| | | | | | A3 - B = A3,MIR | % | ***** | | ***** | | |
| | | | | | IF NOT ADV THEN A3-1=A3,MIR | % | ***** | | ***** | | |
| | | | | | IF NOT LC1 THEN SKIP | % | ***** | | ***** | | |
| | | | | | OPR36X1 -1 = MPCR | % | ***** | | ***** | | |
| | | | | | OUTPUT-1=MPCR | % | ***** | | ***** | | |
| | | | | | ***** | % | ***** | | ***** | | |
| | | | | | YFETCH-1 = CPCR | % | ***** | | ***** | | |
| | | | | | GETA3-1=CPCR | % | ***** | | ***** | | |
| | | | | | MR1; IF RDC | % | ***** | | ***** | | |
| | | | | | WHEN RDC THEN BEX | % | ***** | | ***** | | |
| | | | | | B = A3,BMI | % | ***** | | ***** | | |
| | | | | | A3 - B = A3,MIR | % | ***** | | ***** | | |
| | | | | | IF NOT ADV THEN A3-1=A3,MIR | % | ***** | | ***** | | |
| | | | | | IF NOT LC1 THEN SKIP | % | ***** | | ***** | | |
| | | | | | OPR36X1 -1 = MPCR | % | ***** | | ***** | | |
| | | | | | OUTPUT-1=MPCR | % | ***** | | ***** | | |
| | | | | | ***** | % | ***** | | ***** | | |
| | | | | | YFETCH-1 = CPCR | % | ***** | | ***** | | |
| | | | | | GETA3-1=CPCR | % | ***** | | ***** | | |
| | | | | | MR1; IF RDC | % | ***** | | ***** | | |
| | | | | | WHEN RDC THEN BEX | % | ***** | | ***** | | |
| | | | | | B = A3,BMI | % | ***** | | ***** | | |
| | | | | | A3 - B = A3,MIR | % | ***** | | ***** | | |
| | | | | | IF NOT ADV THEN A3-1=A3,MIR | % | ***** | | ***** | | |
| | | | | | IF NOT LC1 THEN SKIP | % | ***** | | ***** | | |
| | | | | | OPR36X1 -1 = MPCR | % | ***** | | ***** | | |
| | | | | | OUTPUT-1=MPCR | % | ***** | | ***** | | |
| | | | | | ***** | % | ***** | | ***** | | |
| | | | | | YFETCH-1 = CPCR | % | ***** | | ***** | | |
| | | | | | GETA3-1=CPCR | % | ***** | | ***** | | |
| | | | | | MR1; IF RDC | % | ***** | | ***** | | |
| | | | | | WHEN RDC THEN BEX | % | ***** | | ***** | | |
| | | | | | B = A3,BMI | % | ***** | | ***** | | |
| | | | | | A3 - B = A3,MIR | % | ***** | | ***** | | |
| | | | | | IF NOT ADV THEN A3-1=A3,MIR | % | ***** | | ***** | | |
| | | | | | IF NOT LC1 THEN SKIP | % | ***** | | ***** | | |
| | | | | | OPR36X1 -1 = MPCR | % | ***** | | ***** | | |
| | | | | | OUTPUT-1=MPCR | % | ***** | | ***** | | |
| | | | | | ***** | % | ***** | | ***** | | |
| | | | | | YFETCH-1 = CPCR | % | ***** | | ***** | | |
| | | | | | GETA3-1=CPCR | % | ***** | | ***** | | |
| | | | | | MR1; IF RDC | % | ***** | | ***** | | |
| | | | | | WHEN RDC THEN BEX | % | ***** | | ***** | | |
| | | | | | B = A3,BMI | % | ***** | | ***** | | |
| | | | | | A3 - B = A3,MIR | % | ***** | | ***** | | |
| | | | | | IF NOT ADV THEN A3-1=A3,MIR | % | ***** | | ***** | | |
| | | | | | IF NOT LC1 THEN SKIP | % | ***** | | ***** | | |
| | | | | | OPR36X1 -1 = MPCR | % | ***** | | ***** | | |
| | | | | | OUTPUT-1=MPCR | % | ***** | | ***** | | |
| | | | | | ***** | % | ***** | | ***** | | |
| | | | | | YFETCH-1 = CPCR | % | ***** | | ***** | | |
| | | | | | GETA3-1=CPCR | % | ***** | | ***** | | |
| | | | | | MR1; IF RDC | % | ***** | | ***** | | |
| | | | | | WHEN RDC THEN BEX | % | ***** | | ***** | | |
| | | | | | B = A3,BMI | % | ***** | | ***** | | |
| | | | | | A3 - B = A3,MIR | % | ***** | | ***** | | |
| | | | | | IF NOT ADV THEN A3-1=A3,MIR | % | ***** | | ***** | | |
| | | | | | IF NOT LC1 THEN SKIP | % | ***** | | ***** | | |
| | | | | | OPR36X1 -1 = MPCR | % | ***** | | ***** | | |
| | | | | | OUTPUT-1=MPCR | % | ***** | | ***** | | |
| | | | | | ***** | % | ***** | | ***** | | |
| | | | | | YFETCH-1 = CPCR | % | ***** | | ***** | | |
| | | | | | GETA3-1=CPCR | % | ***** | | ***** | | |
| | | | | | MR1; IF RDC | % | ***** | | ***** | | |
| | | | | | WHEN RDC THEN BEX | % | ***** | | ***** | | |
| | | | | | B = A3,BMI | % | ***** | | ***** | | |
| | | | | | A3 - B = A3,MIR | % | ***** | | ***** | | |
| | | | | | IF NOT ADV THEN A3-1=A3,MIR | % | ***** | | ***** | | |
| | | | | | IF NOT LC1 THEN SKIP | % | ***** | | ***** | | |
| | | | | | OPR36X1 -1 = MPCR | % | ***** | | ***** | | |
| | | | | | OUTPUT-1=MPCR | % | ***** | | ***** | | |
| | | | | | ***** | % | ***** | | ***** | | |
| | | | | | YFETCH-1 = CPCR | % | ***** | | ***** | | |
| | | | | | | | | | | | |

| | | | | | | | |
|------|------|------|------|------|------|---------------------------|----------|
| 0480 | 0C00 | 0000 | 0000 | 0000 | C060 | YFETCH-1 = CPCR | 12460C00 |
| 0481 | 2490 | 0000 | 0000 | 0000 | 0060 | GETAB-1=CPCR | 12470C00 |
| 0482 | A809 | 00C0 | 0000 | 0000 | C8F0 | MR1; IF RDC | 12480C00 |
| 0483 | AC08 | 0000 | 0C00 | 00F0 | | WHEN RDC THEN BEX | 12490C00 |
| 0484 | 4809 | EC40 | 0030 | 00F0 | | A3 + B = MIR | 12500C00 |
| 0485 | 7C09 | EC46 | 0030 | 00F0 | | IF AOV THEN A3+B+1=MIR | 12510000 |
| 0486 | 6C09 | 00C0 | 0030 | 00F0 | | IF ABT THEN O=MIR | 12520C00 |
| 0487 | 22F0 | 00C0 | 0000 | 0040 | | OUTPUT-1=MPCR | 12530C00 |
| | | | | | | % ***** | 12540000 |
| | | | | | | OPR15: | 12550C00 |
| 0488 | 0C00 | 0000 | 0C00 | 0060 | | YFETCH-1 = CPCR | 12560C00 |
| 0489 | 24C0 | 0000 | 0030 | 0060 | | GETAA3-1=CPCR | 12570C00 |
| 048A | A809 | 00C0 | 0000 | 00F0 | | MR1; IF RDC | 12580C00 |
| 048B | AC08 | EDC0 | 10C0 | 00F0 | | WHEN RDC THEN A3+1=A3,BEX | 12590C00 |
| 048C | 4839 | E155 | 003C | 00F0 | | A3 AND LIT=HAR | 12600C00 |
| 048D | 007C | 00C0 | 000C | 00E0 | | 7=LIT | 12610000 |
| 048E | 4809 | 0C40 | 103C | 00F0 | | B = A3,BMI | 12620C00 |
| 048F | 4809 | EC40 | 0030 | 00FC | | A3 + B = MIR | 12630C00 |
| 0490 | 7C09 | EC46 | 0030 | 00F0 | | IF AOV THEN A3+B+1=MIR | 12640C00 |
| 0491 | 6C09 | 0000 | 0C30 | 00F0 | | IF ABT THEN O=MIR | 12650C00 |
| 0492 | 22F0 | 0000 | 0000 | 0040 | | OUTPUT-1=MPCR | 12660C00 |
| | | | | | | % ***** | 12670C00 |
| | | | | | | OPR16: | 12680C00 |
| 0493 | 0C0C | 0000 | 0000 | 0060 | | YFETCH-1 = CPCR | 12690C00 |
| 0494 | 249C | 0000 | 0000 | 0060 | | GETAB-1=CPCR | 12700000 |
| 0495 | 4809 | EDC0 | 0030 | 00F0 | | NOT A3 = MIR | 12710C00 |
| 0496 | 22F0 | 0000 | 0030 | 0040 | | OUTPUT-1=MPCR | 12720C00 |
| | | | | | | % ***** | 12730C00 |
| | | | | | | OPR17: | 12740C00 |
| 0497 | 0C00 | 0000 | 0030 | 0060 | | YFETCH-1=CPCR | 12750000 |
| 0498 | 2490 | 0000 | 0030 | 0060 | | GETAB-1=CPCR | 12760C00 |
| 0499 | 4809 | E000 | 0C30 | 00F0 | | A3 | 12770C00 |
| 049A | 4C09 | E002 | 0C30 | 00F0 | | IF MST THEN NOT A3 = MIR | 12780C00 |
| 049B | 22F0 | 0000 | 0000 | 0040 | | OUTPUT-1=MPCR | 12790C00 |
| | | | | | | % ***** | 12800C00 |
| | | | | | | OPR20: | 12810000 |
| 049C | 0C00 | 0000 | 0030 | 006C | | YFETCH-1 = CPCR | 12820C00 |
| 049D | 24CC | 0000 | 0030 | 0060 | | GETAA3-1=CPCR | 12830C00 |
| 049E | 4809 | E002 | 0000 | 00F0 | | NOT A3 | 12840C00 |
| 049F | 0A5C | 0000 | 0000 | 00C0 | | IFETCH-1=AMPCR | 12850C00 |
| 04A0 | 6829 | 0000 | 0000 | 00F0 | | IF ABT THEN JUMP | 12860000 |
| 04A1 | 4809 | E140 | 003C | 00F0 | | A3 + LIT = HAR | 12870C00 |
| C4A2 | 0080 | 0000 | 0C0C | 00E0 | | BBASE = LIT | 12880C00 |
| C4A3 | 22F0 | 0000 | 0030 | 0040 | | OUTPUT-1=MPCR | 12890C00 |
| | | | | | | % ***** | 12900C00 |
| | | | | | | OPR21: | 12910000 |
| 04A4 | 0C0C | 0000 | 000C | C060 | | YFETCH-1 = CPCR | 12920C00 |
| 04A5 | 24C0 | 00C0 | 003C | 0060 | | GETAA3-1=CPCR | 12930C00 |
| 04A6 | 4809 | E002 | 0000 | 00F0 | | NOT A3 | 12940C00 |
| 04A7 | 0A50 | 0000 | 0000 | 00C0 | | IFETCH-1=AMPCR | 12950C00 |
| C4A8 | 6829 | 00C0 | 0030 | 00F0 | | IF ABT THEN JUMP | 12960C00 |
| 04A9 | 4809 | E140 | 003C | 00F0 | | A3 + LIT = HAR,BMI | 12970C00 |
| 04AA | 008C | 0000 | 0C0C | 00A0 | | BBASE = LIT; 16 = SAR | 12980C00 |
| 04AB | A809 | 0000 | 0000 | 00F0 | | MR1; IF RDC | 12990C00 |
| 04AC | AC08 | 0C40 | 10C0 | 00F0 | | WHEN RDC THEN B = A3,BEX | 13000C00 |
| 04AD | 4809 | EC40 | 0030 | 00F0 | | A3+B=MIR | 1301000 |

116


```

0546 4809 2C56 0000 80C0
0547 4809 E0C0 9000 C0F0
0548 4849 E000 0000 00F0
0549 5C19 D8F0 20C0 00F0
054A 4809 D81C 2020 00F0
054B 0A50 0000 0C3C C040

054C 0CFC 0000 0030 0060
054D 249C 0000 0000 0060
054E 4809 0C52 C000 00F0
054F 6019 00C0 0000 00FC
0550 0A50 0000 0070 C040
0551 4809 2C40 0C3C 00F0
0552 0080 0000 0C00 00A0
0553 4809 C0C1 AC00 08F0
0554 A000 0000 0030 0030
0555 AC08 00C0 0C00 00F0
0556 4809 0C40 1D00 40F0
0557 4809 E0C0 0C30 C0F0
0558 4849 EC4C 0000 00F0
0559 4808 EC5E 0000 C0F0
055A 77D8 00C0 0C30 00F0
055B 7FC9 00C0 003C 00F0
055C 2C19 D81D AC00 C0F0
055D 4809 D80F A000 00F0
055E 22FC 00C0 0000 0040

055F 0CFC 0000 0030 0060
0560 249C 0000 0000 0060
0561 49C9 E000 0030 00F0
0562 A809 0030 0C00 08F0
0563 AC08 0000 0C0C 00F0
0564 4809 0C40 1D3C C0F0
0565 286C 00C0 0000 0040

0566 0CFC 00C0 003C 0060
0567 2490 0000 0020 0060
0568 A809 0000 0030 08F0
0569 0A50 00C0 0C3C C0C0
056A AC08 0F46 0C2C 00F0
056B 4809 2F56 000C 00F0
056C A809 0C40 4000 C8F0
056D AFC8 0C00 0C30 00F0
056E 4849 0C40 1C00 00F0
056F 4809 A000 0D30 00F0
0570 0F1C 0000 0020 0040

0571 0CFC 0000 0030 0060
0572 249C 0000 0030 C060
0573 A809 0000 0070 C8F0
0574 AFC8 0F46 0C3C 00F0
0575 4809 2F56 000C C0F0
0576 A809 EC56 1090 08F0
0577 AC08 0000 0C00 00F0

LIT AND B=SAR
A3 R=A3
A3 SET LC2
IF LST THEN A2 AND B011 = A2 SKIP
A2 OR B10C=A2
IFETCH-1=MPCR

*****
OPR43:
COMPARE INDEX INCREMENT
%CI
%Y VALUE =A3,MIR
%RETURNS A-FIELD IN B&MAR
%TEST FOR F(0) => NOOP IF TRUE

% B(A) ADDRESS
%GET LIMITS BIT TO MSB
%READ B(A)
%A3=B(A)/B=C(A)
%INCREMENT B(A) THEN TEST B(A)>A(A)
%TEST FOR UNLIKE SIGNS/SIGNAL REPEAT
A3 GEO B/ IF MST THEN STEP ELSE SKIP
SKIP/ IF FALSE THEN 0=MIR/SET LC1 %SET B(A)=0
IF TRUE THEN 0=MIR/SET LC1 %SET B(A) = 0
IF LC1 THEN A2 OR B100 C=A2/SKIP %SET OUT OF LIMITS
A2 AND B011 C=A2
%CLEAR LIMITS BIT
OUTPUT-1=MPCR

*****
OPR44:
COMPARE
%CI
%Y VALUE = A3,MIR
%RETURNS A-FIELD IN B&MAR
MR1/IF RDC
WHEN RDC THEN BEX
B=A3,BMI,CSAR
A3+1=MIR
A3 XOR B/SET LC2
A3 GEO B/ IF MST THEN STEP ELSE SKIP
SKIP/ IF FALSE THEN 0=MIR/SET LC1 %SET B(A)=0
IF TRUE THEN 0=MIR/SET LC1 %SET B(A) = 0
IF LC1 THEN A2 OR B100 C=A2/SKIP %SET OUT OF LIMITS
A2 AND B011 C=A2
%CLEAR LIMITS BIT
OUTPUT-1=MPCR

*****
OPR45:
COMPARE LIMITS
%CI
%Y BETWEEN A(A) AND A(A+1) OR = A(A) THEN CD WITHIN LIMITS
% ELSE OUTSIDE OF LIMITS
YFETCH-1 = CPCR
GETAB-1=CPCR
MR1/IF RDC
IFETCH-1=AMPCR
WHEN RDC THEN B&MAR+1=MAR,BEX
LIT AND B&MAR=MAR
B=A1,MIR/IF RDC
WHEN RDC THEN BEX/SET LC1
B=A3/SET LC2
A1=MIR,BMI
OPR4XX5-1=MPCR

*****
OPR46:
COMPARE MASKED
%CH
%Y VALUE => A3,MIR
%RETURNS A-FIELD IN B&MAR
MR1/ IF RDC
WHEN RDC THEN B&MAR+1=MAR,BEX/SET LC1
LIT AND B&MAR=MAR
A3 AND B = A3,MIR/MR1/ IF RDC
WHEN RDC THEN BEX
% A(A+1) => B

```



```

0578 4809 0C40 1020 00F0      B = A3,BM1
0579 2860 0000 0000 0040      SETCOMPARE-1=MPCR
                                % *****
                                % COMPARE GATED
                                % CG
                                % A3=(Y+1),B=(A)AND(Y)
                                % A3=A(A) AND Y/ B=(A+1)
                                % *****
OPR47:
057A 0CF0 0000 0030 0060      YFETCH-1 = CPCR
057B 2490 0000 0030 0060      GETAB-1=CPCR
057C A809 0000 0030 08F0      MR1/ IF RDC
                                % *****
                                % WHEN RDC THEN BHAR+1=HAR,DEX,SET LCI
                                % LIT AND BHAR=HAR
                                % IF NOT ADV THEN A3-B-1=A3,B
                                % A3-B=A3,B,MR1/IF RDC
                                % IF NOT ADV THEN A3-1=A3,B
                                % IF HST THEN NOT B=A3
                                % WHEN RDC THEN BEX
                                % SETCOMPARE-1=MPCR
                                % *****
                                % JUMP PARITY / DOUBLE PRECISION
                                % *****
OPR50:
0585 1270 0000 0030 0060      Y2FETCH-1=CPCR
0586 24FC 0000 0000 0060      GETF1-1=CPCR
0587 4809 E640 0040 C0F0      A3+AMPCR=AMPCR
0588 068C 00C0 0030 00C0      OPR50XX-1=AMPCR
0589 4809 A0C0 8800 00F0      A1 R=B
058A 8070 00C0 0030 00A0      23=SAR,7=LIT
058B 4824 0F40 4000 C0F0      BHAR=A1,EXEC
                                % *****
                                % LIT AND B=HAR
                                % MR1/ IF RDC
                                % WHEN RDC THEN BHAR+1=HAR,BEX
                                % LIT AND BHAR=HAR,JUMP
                                % A3=Y-ADDRESS/ B=(A(A))
                                % *****
                                % JUMP ON CONDITIONS
                                % *****
OPR51:
0590 1270 0000 0030 0060      Y2FETCH-1=CPCR
0591 24F0 0000 0030 0060      GETF1-1=CPCR
0592 4809 E640 0040 C0F0      A3+AMPCR=AMPCR
0593 0700 0000 0030 00C0      OPR51XX-1=AMPCR
0594 4809 0F40 1030 C0F0      BHAR=A3
0595 4809 A0C0 8800 00F0      A1 R=B
0596 8070 0000 0030 00A0      23 = SAR, 7 = LIT
0597 4809 2C56 000C 00F0      LIT AND B=HAR
0598 A824 0000 0000 08F0      EXEC,MR1/IF RDC
0599 AC28 00C0 0C00 C0F0      WHEN RDC THEN BEX,JUMP
                                % *****
                                % LEAVES (A) IN B/ JUMP ADDRESS IN A3
                                % JUMP AND INDEX
                                % *****
OPR52:
059A 1270 0000 0030 0060      Y2FETCH-1=CPCR
059B 24FC 0000 0030 0060      GETF1-1=CPCR
059C 4809 E640 0040 00F0      A3+AMPCR=AMPCR
059D 0780 0000 0030 00C0      OPR52XX-1=AMPCR
059E 4809 0F40 0800 00F0      BHAR=B
059F 4809 A000 9000 00F0      A1 R=A3
05A0 3070 0000 0030 00A0      23=SAR,7=LIT
05A1 4809 E156 5030 00F0      A3 AND LIT = A3,A1, MR1
05A2 4809 E140 003C 00F0      A3+LIT=HAR
05A3 0080 0000 0030 00E0      BBASE=LIT
05A4 A824 0000 0000 08F0      EXEC,MR1/IF RDC
05A5 AC28 0C40 1C30 00F0      WHEN RDC THEN B=A3,BEX,JUMP
                                % *****
                                % B=(B(A))/ A3=JUMP ADDRESS
                                % JUMP ON CD
                                % *****
OPR53:
05A6 1270 0000 0000 0060      Y2FETCH-1=CPCR
                                % *****
                                % JUMP ADDR. IN BHAR
                                % *****

```



```

OPR65:
C60B 2600 0000 0000 0060
C60C 4809 0F46 0C9C 00FC
C60D 4809 2F56 000C 00F0
C60E 0070 0C00 0030 C0E0
C60F 2019 00C0 0020 00FC
C610 0F3C 0C00 0000 0040
C611 4809 0C40 900C 08F0
C612 4C08 0000 0C00 00F0
C613 4809 0C40 8C9C 00F0
C614 9809 CC41 0B30 18F0
C615 9E08 EC5C 0030 20F0
C616 4809 CBC2 8C9C 00F0
C617 0000 0000 0030 0030
C618 0B8C 0000 0C00 C040
C619 4809 0000 CC90 08F0
C61A 4C08 0000 0C30 C0F0
C61B 9809 0000 0030 18F0
C61C 9C08 0C40 8090 00F0
C61D 4809 CBC2 809C C0F0
C61E 0000 0C00 0030 0030
C61F 0B80 0C00 0000 C040

C620 2600 0000 CC9C 0060
C621 4809 0C40 8080 40F0
C622 0B80 0000 0030 00CC
C623 4C00 0019 0F0C 00F0
C624 4820 0C40 0030 C0F0

C625 2600 00C0 0030 0060
C626 4809 0F46 009C 00F0
C627 4809 2F56 000C 00F0
C628 007C 0000 0000 00E0
C629 2019 0000 0000 00F0
C62A 0F40 0000 000C 0040
C62B 4809 0C40 9C00 08F0
C62C 4C08 0000 0C0C 00F0
C62D 4809 0C40 8090 C0F0
C62E 4C08 0C19 0F00 C0FC
C62F 4809 CC40 0C90 C0F0
C630 9809 0C41 0B30 18F0
C631 9E08 EC5C 0030 20F0
C632 4809 0B02 809C 00F0
C633 0000 0C00 0030 0030
C634 0B80 0000 0030 C040

C635 4809 0000 0000 C8F0
C636 4C08 0000 1C00 00F0
C637 4809 0C40 8C90 00F0
C638 4C08 00C2 1000 00F0
C639 481B 0019 0F00 00F0
C63A 4809 0000 0000 C0F0
C63B 4809 E0C0 0090 C0FC
C63C 9809 0000 000C 18F0
C63D 9C08 0B02 8C9C 00F0

OPR65:
EXAMPLE INPUT A+1=ZZWW, A=XXYY; OUTPUT A+1=00ZZ, A=WWXX
SHIFTAMOUNT-1=CPCR
BMAR+1=MAR
%B=A-RCG;MAR=A-A00R;SAR=SHIFT AMOUNT
LIT AND BMAR=MAR
%CONVERT A(8) TO A(0)
7=LIT
IF NOT LC1 THEN SKIP
OPR65XX-1=MPCR
B R=A3;MR1;IF R0C
WHEN RDC THEN BEX
B R=MIR;BEX
B L=B;MW1;IF SAI
WHEN SAI THEN A3 OR 0=MIR;ASR
NOT CTR R=MAR
24=SAR
%RESET A-ADDRESS
%B=WWXX
HALFFETCH-1=MPCR
OPR65XX: 0=MIR;MR1;IF RDC
WHEN RDC THEN BEX
MW1;IF SAI
WHEN SAI THEN B R=MIR
NOT CTR R=MAR
24=SAR
HALFFETCH-1=MPCR
***** SHIFT RIGHT SIGN FILL *****
OPR66: %SAR=SHIFT AMOUNT;MAR=AIREG A00R; B=(A)
SHIFTAMOUNT-1=CPCR
B R=MIR;CSAR
HALFFETCH-1=AMPCR
IF MST THEN B111 L=BBI;STEP ELSE JUMP
B=MIR;JUMP
***** SHIFT RIGHT DBLE/FILL SIGN *****
OPR67:
EXAMPLE INPUT A+1=ZZWW,A=XXYY; OUTPUT=A+1=00ZZ,A=WWXX
EXAMPLE INPUT=-ZZWW,A=XXYY; OUTPUT=A+1=FFZZ, A=WWXX
SHIFTAMOUNT-1=CPCR
BMAR+1=MAR
LIT AND BMAR=MAR
%CONVERT A(8) TO A(0)
7=LIT
IF NOT LC1 THEN SKIP
OPR67XX-1=MPCR
B R=A3;MR1;IF R0C
WHEN RDC THEN BEX
B R=MIR
IF MST THEN B111 L=BBI; STEP ELSE SKIP %CHECK FOR NEGATIVE
B=MIR;BEX
B L=B;MW1;IF SAI
WHEN SAI THEN A3 OR B=MIR;ASR %MIR=WWXX
NOT CTR R=MAR
%RESET A-ADDRESS
%OUTPUT=WWXX
24=SAR
%AA+1=SIGC/ZZ
HALFFETCH-1=MPCR
%B=(A);SAR=SHIFT-32
MR1;IF RDC
WHEN RDC THEN 0=A3;BEX
B R=MIR
IF MST THEN NOT 0=A3;STEP ELSE SKIP
B111 L=BBI;SKIP
A3=MIR
MW1;IF SAI
WHEN SAI THEN NOT CTR R=MAR

```


| | | | | | | | |
|------|-------|------|------|------|-----------------|--|------------|
| 0670 | 4820 | 0000 | 0000 | 00F0 | JUMP | X JUMP TO SUBOPCODE IN AMPCR | 1846C000 D |
| | | | | | X ***** | INCLUSIVE OR (A OR) | 18470000 D |
| 0671 | 4809' | EC5C | 0030 | 00F0 | OPR01X0: | X INCLUSIVE Y OR A => A | 1848C000 D |
| 0672 | 2410 | 0000 | 0000 | 0040 | | | 18490000 D |
| | | | | | X ***** | | 18500C00 D |
| | | | | | OUTLOGIC-1=MPCR | | 18510C00 D |
| | | | | | OPR01X1: | SELECTIVE CLEAR A WITH Y=>A(A+1) ***** | 18520000 D |
| | | | | | X ***** | | 18530000 D |
| 0673 | 4809 | EC44 | 0030 | 00F0 | | | 18540000 D |
| 0674 | 2410 | 0000 | 0000 | 0040 | | | 1855C000 D |
| | | | | | X ***** | | 18560000 D |
| | | | | | OPR01X2: | SELECTIVE SUBSTITUTE ***** | 18570C00 D |
| | | | | | X ***** | | 18580C00 D |
| 0675 | 4809 | 0F46 | 000C | 00F0 | | | 18590C00 D |
| 0676 | 4809 | 2F56 | 000C | 00F0 | | | 18600C00 D |
| 0677 | A809' | E000 | 0030 | C8F0 | | | 1861C000 D |
| 0678 | AC08 | 0C42 | 1C00 | 00F0 | | | 1862C000 D |
| | | | | | X ***** | | 18630000 D |
| 0679 | 4809 | EC56 | 0030 | 00F0 | | | 18640C00 D |
| | | | | | X ***** | | 18650C00 D |
| 067A | 4809 | EC44 | 0F00 | 00F0 | | | 18660C00 D |
| 067B | 4809 | 0C40 | 0030 | 00F0 | | | 18670C00 D |
| 067C | 2410 | 0000 | 0000 | 0040 | | | 18680C00 D |
| | | | | | X ***** | | 18690C00 D |
| | | | | | OPR01X3: | EXCLUSIVE OR A XOR Y ***** | 18700000 D |
| | | | | | X ***** | | 18710C00 D |
| 067D | 4809 | EC4C | 0030 | 00F0 | | | 18720000 D |
| 067E | 2410 | 0000 | 0000 | 0040 | | | 1873C000 D |
| | | | | | X ***** | | 1874C000 C |
| | | | | | OPR01X4: | AND A(A+1)+A(A) AND Y => A(A+1) ***** | 18750C00 D |
| | | | | | X ***** | | 1876C000 D |
| 067F | 4809 | 0F46 | 003C | 00F0 | | | 1877C000 D |
| 0680 | 4809 | 2F56 | 003C | 00F0 | | | 18780C00 D |
| 0681 | A809 | EC56 | 1000 | 08F0 | | | 18790000 D |
| 0682 | AC08 | 0000 | 0C3C | 00F0 | | | 1880C000 D |
| 0683 | 4809 | EC40 | 0030 | 00F0 | | | 18810C00 D |
| 0684 | 7C09 | EC46 | 0030 | 00F0 | | | 18820C00 D |
| 0685 | 6C09 | 0000 | 0030 | 00F0 | | | 18830C00 D |
| 0686 | 2410 | 0000 | 000C | 0040 | | | 18840C00 D |
| | | | | | X ***** | | 18850C00 D |
| | | | | | OPR01X5: | AND A(A) AND Y => A(A) ***** | 18860C00 D |
| | | | | | X ***** | | 18870C00 D |
| 0687 | 4809 | EC56 | 0030 | 00F0 | | | 18880C00 D |
| 0688 | 22F0 | 0000 | 0030 | 0040 | | | 18890C00 D |
| | | | | | X ***** | | 1890C000 D |
| | | | | | OPR01X6: | AND SUBTRACT A(A+1)-Y AND A(A) => A(A+1) ***** | 1891C000 D |
| | | | | | X ***** | | 18920000 D |
| 0689 | 4809 | 0F46 | 000C | 00F0 | | | 18930C00 D |
| 068A | 4809 | 2F56 | 003C | 00F0 | | | 18940C00 D |
| 068B | A809 | EC56 | 0030 | 08F0 | | | 18950C0C D |
| 068C | AC08 | 0000 | 0C00 | 00F0 | | | 18960C00 D |
| 068D | 4809 | 0C40 | 1000 | 00F0 | | | 18970C00 D |
| 068E | 4809 | EC5E | 0030 | 00F0 | | | 18980C00 D |
| 068F | 7409 | EC58 | 0030 | 00F0 | | | 18990C00 D |
| 0690 | 2410 | 0000 | 0000 | 0040 | | | 19000C00 D |
| | | | | | X ***** | | 19010C00 D |
| | | | | | OPR01X7: | AND A(A) AND Y => A(A+1) ***** | 19020C00 D |
| | | | | | X ***** | | 19030C00 D |
| 0691 | 4809 | 0F46 | 003C | 00F0 | | | 19040C00 D |
| 0692 | 4809 | 2F56 | 000C | 00F0 | | | 19050C00 D |
| 0693 | 4809 | EC56 | 0030 | 00F0 | | | |


```

C6C2 A809 0C4D 0030 C8FC B=MIR,BMI,MR1,IF RDC %MIR=(A),B=Y-ADDR 19660C00 D
C6C3 AC08 0C4D 0C1C 00F0 WHEN RDC THEN B=MAR2,BEX %B=(A(A+1)),MAR2=Y 1967CC00 D
C6C4 9809 00C0 0030 1CF0 MW2,IF SAI 19680C00 D
C6C5 9C08 0F46 001C 00F0 WHEN SAI THEN BMAR+1=MAR2 1969CC00 D
C6C6 4809 0C40 0C30 00F0 B=MIR %SETUP A+1 FOR OUTPUT 1970CC00 D
C6C7 2320 C000 0030 0040 OUTPUT2NORET-1=MPCR 19710C00 C
% ***** TEST ANC SET FLAG ***** 1972CC00 D
OPR03X7: Y2FETCH-1=CPCR 19730C00 C
B %BMAR=YADDRESS, B=(Y) 19740C00 D
IF NOT M5T THEN BATT=MIR,SKIP %TEST MOST SIGNIFICANT BIT 1975CC00 D
OPR03XX7-1=MPCR 19760C00 D
MW2,IF SAI 19770C00 D
WHEN SAI THEN A2 OR B100=A2,SKIP 1978CC00 C
OPR03XX7: A2 AND B011=A2 1979CC00 D
IFETCH-1=MPCR 19800C00 D
% ***** DOUBLE LOAD A ***** 19810C00 D
OPR05X0: %MIR=(Y),MAR2=YADDR %READ Y+1 *****19820C00 C
BMAR+1=MAR2 19830C00 C
MR2,IF RDC 1984CC00 D
WHEN RDC THEN A3=MAR,BEX %A3=A-ADDRESS 1985CC00 D
MW3,IF RDC %WRITE Y 19860C00 D
WHEN SAI THEN A3+1=MAR 19870C00 D
LIT AND BMAR=MAR %CONVERT A(8) TO A(0) 19880C00 D
B=MIR 19890C00 D
OUTPUT-1=MPCR %WRITE Y+1 19900C00 D
% ***** DOUBLE ADD ***** 19910C00 D
OPR05X1: %A3=A-ADDRESS,MAR2=Y-ADDRESS,MIF=(Y) ***** 19920C00 D
BMAR+1=MAR2,BMI 19930C00 D
B=A1,MR2,IF RDC %A1=(Y) 19940C00 D
WHEN RDC THEN A3=MAR,BEX 19950C00 D
B=A3,MR1,IF RDC %A3=(Y+1) 19960C00 D
WHEN RDC THEN BEX %B=(A) 1997CC00 D
A1+B=A1,MIR %A1=A+Y 19980C00 D
BMAR=B, IF ADV THEN SET LC1 %TEST FOR CARRY 19990C00 D
BMAR+1=MAR 20000C00 D
LIT AND BMAR = MAR %CONVERT A(8) TO A(0) 2001CC00 D
MR1,IF RDC 20020000 D
WHEN RDC THEN B=MAR,BEX 2003CC00 D
IF LC1 THEN A3+B=A3,SKIP %ADD CARRY 20040C00 D
A3+B=A3 20050C00 D
IF ADV THEN A1+1=A1,MIR %ADD CARRY 20060000 D
NOT A3 20070C00 D
IF ABT THEN NOT A1,STEP ELSE SKIP 20080C00 D
IF ABT THEN O=A1,A3,MIR %OUTPUT A+Y 2009CC00 D
MW1,IF SAI 20100C00 D
WHEN SAI THEN BMAR+1=MAR %CONVERT A(8) TO A(0) 20110C00 D
LIT AND BMAR=MAR 20120C00 D
A3=MIR 20130C00 D
OUTPUT-1=MPCR %OUTPUT (A+1)*(Y+1) 2014CC00 D
% ***** DOUBLE SUBTRACT (A+1,A)-(Y+1,Y)=>A+1,A ***** 2015CC00 D
OPR05X2: %A3=A-ADDR,MAR2=Y-ADDR,MIR=(Y) 20160C00 D
BMAR+1=MAR2,BMI %B=(Y) 20170C00 C
B=A1,MR2,IF RDC %A1=(Y) 20180C00 D
WHEN RDC THEN A3+1=MAR,BEX 20190C00 C
LIT AND BMAR=MAR %CONVERT A(8) TO A(0) 20200C00 D
B=MIR,MR1,IF RDC %MIR=(Y+1),READ A+1 20210C00 D
WHEN RDC THEN A3=MAR,BEX %B=(A+1) 20220000 D
B=A3,BMI %A3=(A+1) 2023CC00 D
A3-B=A3,MR1,IF RDC %A3=(A+1)-(Y+1) 2024CC00 D
20250C00 D

```



```

06F6 73C9 0000 0000 C0F0 IF NOT ADV THEN SET LC1
06F7 AC08 A0C0 0C30 C0F0 WHEN RDC THEN A1=MIR,BEX XB=(A)
06F8 4809 0C40 40C0 00F0 B=A1,BMI
06F9 2C19 AC58 4030 C0F0 IF LC1 THEN A1-B=A1,MIR,SKIP
06FA 4809 AC5E 4030 00F0 A1-B=A1,MIR
06FB 7408 E0DE 1000 00F0 IF NOT ADV THEN A3-1=A3,STEP ELSE SKIP
06FC 7409 A0DE 4030 00FC IF NOT ADV THEN A1-1=A1,MIR
06FD 9809 00C0 0C00 18F0 HW1,IF SAI
06FE 9C08 0F46 000C 00F0 WHEN SAI THEN BHAR+1=HAR WRITE A
06FF 4809 2F56 000C 00F0 LIT AND BHAR=HAR
0700 4809 E0C0 0030 00F0 A3=MIR
0701 22FC 0000 0C0C 004C OUTPUT-1=MPCR
% ***** DOUBLE COMPARE *****
OPR05X3:
7302 4809 0F46 001C C0F0 BHAR+1=HAR2
7303 8009 C0C1 A030 0CFC A2 C=A2,BMI,MR2,IF RDC
7304 800C 00C0 0C00 0030 COMP 1=SAR
7305 4809 0C40 4000 00F0 B=A1
7306 AC08 E0C0 0C0C 00F0 WHEN RDC THEN A3=HAR,BEX
7307 8009 0C40 10C0 08F0 B=A3,MR1,IF RDC
7308 AC08 0F46 0C0C 00F0 WHEN RDC THEN BHAR+1=HAR,BEX XA1=Y,A3=Y+1,B=A
7309 4809 2F56 000C 00F0 LIT AND BHAR=HAR XCONVERT A(8) TO A(O)
730A 8009 AC52 000C 08F0 A1 EOL B,MR1,IF RDC
730B 68C9 AC4C 0000 00F0 A1 XOR B,IF TRUE THEN SET LC1 XTEST SIGNS,LC1 => A EOL Y
730C 0A50 0000 0000 00C0 IFETCH-1=AMPCR
730D 480E AC5E 000C 40F0 A1 LSS B,CSAR,IF MST THEN STEP ELSE SKIP
730E 7449 00C0 0000 00F0 IF FALSE THEN SET LC2 XSIGNS WERE DIFFERENT
730F 7249 0000 0030 00F0 IF TRUE THEN SET LC2
7310 AC08 00C0 0C30 C0F0 WHEN RDC THEN BEX
7311 4809 EC52 0000 00F0 A3 EOL B
7312 640B C0CE 2000 C0F0 IF FALSE THEN A2 AND B110 = A2,STEP ELSE SKIP XCLEAR EQUAL BIT
7313 0F70 00C0 0030 0040 OPR05XX3-1=MPCR
7314 2C0B C00C 2030 00F0 IF LC1 THEN A2 OR B001 =A2,STEP ELSE SKIP XA+1,A=Y+1,Y
7315 4820 081D A000 C0F0 A2 OR B100 C=A2,JUMP XSET GEO BIT
7316 3C19 081C 2030 00F0 IF LC2 THEN A2 OR B100=A2,SKIP XA+1,A GIR Y+1,Y SET GTR BIT
7317 4809 080E 2030 00F0 A2 AND B011=A2 XCLEAR GEO BIT
7318 4820 C0CF A030 00F0 A2 AND B110 C=A2,JUMP XCLEAR EQUAL BIT
OPR05XX3: XLOWER HALVES A&Y NOT EQUAL
7319 4809 EC4C 000C C0F0 A3 XOR B
731A 4819 EC5E 0000 00F0 A3 LSS B,IF MST THEN SKIP
731B 7429 081D A000 00F0 IF TRUE THEN A2 OR B100 C=A2,JUMP XSET GEO BIT
731C 7C29 081D A030 C0F0 IF FALSE THEN A2 OR B100 C=A2,JUMF
731D 4820 080F A030 00F0 A2 AND B011 C=A2,JUMP
% ***** LOAD BASE & MEM.PROT. *****
OPR05X4:
071E 2240 0000 0030 C040 UNWRITTEN-1=MPCR
% ***** XPRINT ERROR AND RETURN TO IFETCH *****
OPR06X0:
071F 4800 0000 0000 C0F0 FLOATING POINT ADD
0720 0A50 0C00 0030 0040 WAIT
% ***** FLOATING POINT SUBTRACT *****
OPR06X1:
0721 480C 0000 0030 00F0 IFETCH-1=MPCR
0722 0A50 0000 0C00 C040 WAIT
% ***** FLOATING POINT MULTIPLY *****
OPR06X2:
0723 4800 0000 0000 00F0 IFETCH-1=MPCR
0724 0A50 0000 0C00 0040 WAIT
% ***** FLOATING POINT DIVIDE *****
OPR06X3:

```


| | | | | | | | | |
|------|------|------|------|------|-----------------------------------|---------------|---------------------------|----------|
| 0725 | 4800 | 0000 | 0030 | 00F0 | WAIT | IFETCH-1=MPCR | FLOATING POINT ADD/ROUND | 20860C00 |
| 0726 | 0A50 | 0000 | 0030 | 0040 | % | OPR06X4: | | 2087C000 |
| 0727 | 4800 | 0000 | 0000 | 00F0 | WAIT | IFETCH-1=MPCR | FLOATING POINT SUBR/ROUND | 2089C000 |
| 0728 | 0A50 | 0000 | 0030 | 0040 | % | OPR06X5: | | 2090C000 |
| 0729 | 4800 | 00C0 | 0000 | 00F0 | WAIT | IFETCH-1=MPCR | | 20910C00 |
| 072A | 0A50 | 0000 | 0000 | 0040 | % | OPR06X6: | | 20930C00 |
| 072B | 4800 | 0000 | 0030 | 00F0 | WAIT | IFETCH-1=MPCR | | 20940C00 |
| 072C | 0A50 | 0000 | 0000 | 0040 | % | OPR06X7: | | 2095C000 |
| 072D | 4800 | 00C0 | 0000 | 00F0 | WAIT | IFETCH-1=MPCR | | 20960C00 |
| 072E | 0A50 | 0000 | 0030 | 0040 | % | OPR07X0: | INTERPROCESSOR INTERRUPT | 20980C00 |
| 072F | 2240 | 0000 | 0000 | 0040 | % | OPR07X1: | UNWRITTEN-1=MPCR | 21050C00 |
| 0730 | 2240 | 0000 | 0000 | 0040 | % | OPR07X2: | UNWRITTEN-1=MPCR | 21080C00 |
| 0731 | 2240 | 0000 | 0030 | 0040 | % | OPR07X3: | UNWRITTEN-1=MPCR | 21100C00 |
| 0732 | 2240 | 0000 | 0000 | 0040 | % | OPR07X4: | UNWRITTEN-1=MPCR | 21140C00 |
| 0733 | 4809 | E012 | 0000 | 00F0 | A3 EQL 0 | | | 21170C00 |
| 0734 | 680B | 0000 | 0000 | 00F0 | IF TRUE THEN STEP ELSE SKIP | | | 21180C00 |
| 0735 | 1F40 | 0000 | 0030 | 0040 | CARDREAD-1=MPCR | | | 21190C00 |
| 0736 | 4809 | E002 | 0000 | 00F0 | A3 EQL 1 | | | 21200C00 |
| 0737 | 680B | 0000 | 0030 | 00F0 | IF TRUE THEN STEP ELSE SKIP | | | 21210C00 |
| 0738 | 1F80 | 0000 | 0030 | 0040 | PRINTLINE-1=MPCR | | | 21220000 |
| 0739 | 1FC0 | 00C0 | 0000 | 0040 | DISKREAD-1=MPCR | | | 21230C00 |
| 073A | 2240 | 0000 | 0030 | 0040 | % | OPR07X5: | UNWRITTEN-1=MPCR | 21240000 |
| 073B | 48C9 | 0000 | 0030 | 00F0 | LHAR, RESET GC2 | | | 21250000 |
| 073C | 00F0 | 00C0 | 0000 | 00E0 | 15 = LIT | | | 21260C00 |
| 073D | A809 | 0000 | 0000 | 00F0 | MR1, IF RDC | | | 21270C00 |
| 073E | AC08 | 0000 | 00C0 | 00F0 | WHEN RDC THEN BEX | | | 21280C00 |
| 073F | 4809 | 0C42 | 0000 | 00F0 | NOI B | | | 21290C00 |
| 0740 | 6C0B | C0C0 | 2030 | 00F0 | IF ABT THEN A2 + 1 = A2 ELSE SKIP | | | 21300C00 |
| 0741 | 0A50 | 00C0 | 0030 | 0040 | IFETCH-1 = MPCR | | | 21310000 |
| 0742 | 4809 | 0C40 | 1030 | 00F0 | B = A3 | | | 21320C00 |
| 0743 | 4809 | E00E | 0030 | 00F0 | A3-1 = MIR | | | 21330C00 |
| 0744 | 2300 | 0000 | 0030 | 0060 | OUTPUT1-1 = CPICR | | | 21340C00 |
| 0745 | 4809 | C003 | 001C | 00FC | A2 = MAR2 | | | 21350000 |
| 0746 | A809 | 0000 | 00C0 | 00C0 | MR2, IF RDC | | | 21360000 |
| 0747 | AC08 | A000 | 0030 | 00F0 | WHEN RDC THEN A1=MIR, LHAR, BEX | | | 21370C00 |
| 0748 | 0180 | 0000 | 0000 | 00E0 | RINST=LIT | | | 21380000 |
| 0749 | 2300 | 0000 | 0030 | 0060 | OUTPUT1-1 = CPICR | | | 21390C00 |
| 074A | 4809 | 0F46 | 000C | 00F0 | BHAR + 1 = MAR | | | 21400C00 |
| | | | | | % | INST= | % SAVE REPEAT | 21410C00 |
| | | | | | % | INST= | % SAVE REPEAT | 21420C00 |
| | | | | | % | INST= | % SAVE REPEAT | 21430C00 |
| | | | | | % | INST= | % SAVE REPEAT | 21440C00 |
| | | | | | % | INST= | % SAVE REPEAT | 21450C00 |


```

0748 4809 0C40 4030 C0F0      B = MIP,A1
074C 230C 0000 0000 0060      OUTPUT1-1 = CPCR
074D 4809 0C40 9000 00F0      B R = A3
074E 200C 00C3 0C00 0030      26 = SAR
074F 3C1C 0000 0C00 0060      CONDCHECK-1 = CPCR
0750 2E99 C001 A00C 40F0      IF LC1 THEN A2 C = A2+CSAR;SET GC2; SKIP
0751 0A4C 0C00 0C00 0040      IFETCH1-1 = MPCR
0752 9000 0000 0000 0020      21 = SAR
0753 1888 0000 0000 00F0      SET GC2; WHEN GC2 THEN STEP
0754 4809 C0D0 A00C 00F0      A2 OR B001 C = A2
0755 300C 00C3 0030 0060      S6YFETCH-1 = CPCR
0756 0A4C 0000 0030 C040      IFETCH1-1 = MPCR

% *****
OPR50X0:  %A1=Y-ADDRESS; A3=A(A)      JUMP ON EVEN PARITY *****
B=A3;MR1;IF RDC      %READ A(A+1) MAR=ADDRESS OF A+1
WHEN RDC THEN BEX      A3 AND B=B
COUNTONES-1=CPCR      %A(A) AND A(A+1) = B
A3      %COUNTONES USES A3 FOR COUNTER
IF LST THEN SKIP      %CHECK LSB FOR 000
IF NOT LC1 THEN SET LC1; STEP ELSE SKIP
IF LC1 THEN SET LC1; SKIP
IFETCH-1=MPCR
A1=A3      %PUT JUMP ADDRESS IN A3 FOR PUTJUMP
PUTJUMP-1=MPCR
%IF 000(LST) AND NOT LC1(OPR5CX0) => NO JUMP
%IF 000(LST) AND LC1(OPR5CX1) => JUMP;SET LC1 FOR PUTJUMP
% *****
OPR50X1:  JUMP ON ODD PARITY *****
%USES OPR5CX0 ROUTINE WITH LC1 SET

% *****
OPR50X0-1=MPCR      JUMP IF A(A) AND A(A+1) EQUAL ZERO *****
OPR50X2:  %A1=JUMP ADDRESS;B=(A(A));MAR=A+1 ADDRESS
0 EQL B;MR1;IF RDC      %TEST A(A) FOR 0
IF FALSE THEN SKIF
WHEN RDC THEN BEX
0 EOL B
IF TRUE THEN SET LC2      %TEST (A(A+1))=0
IF LC2 THEN SKIP      %A(A) & A(A+1)=0
IF LC1 THEN STEP ELSE SKIP
IF NOT LC1 THEN SET LC1; SKIP
IFETCH-1=MPCR
A1=A3
PUTJUMP-1=MPCR
%IF A(A) AND A(A+1) = 0 AND OPR50X2 => JUMP
%IF A(A) AND A(A+1) = 0 AND OPR50X3 => NO JUMP
% *****
OPR50X3:  %USES SAME ROUTINES AS OPR50X2 WITH LC1 SET
SET LC1
OPR50X2-1=MPCR
% *****
OPR51X0:  JUMP (A) POSITIVE
%B=(A)
B
IF NOT MST THEN SET LC1
PUTJUMP-1=MPCR
% *****
OPR51X1:  JUMP (A) NEGATIVE
% *****
% NOT B=B
OPR51X0-1=MPCR
% *****
% SAME AS 51X0 EXCEPT COMPLEMENT B
% *****
% JUMP (A) ZERO
% *****

```



```

0PR51X2:      NOT B
              IF ABT THEN SET LC1
              PUTJUMP-1=MPCR
$ *****
$ ***** JUMP (A) NOT ZERO *****
0PR51X3:
              NOT B
              IF NOT ABT THEN SET LC1
              PUTJUMP-1=MPCR
$ *****
$ ***** LOAD E AND JUMP *****
0PR52X0:      $JUMP ADDR. IN A3;A1=8-FIELD
              A1 EOL 0
              IF FALSE THEN A2=MIR;SET LC1;SKIP $MAR1=ADDR OF E(A)
              IFETCH-1=MPCR
              OUTPUT1-1=CPCR
              PUTJUMP-1=MPCR
$ *****
$ ***** INDEX JUMP B *****
0PR52X1:      $B=(B(A)); A1=P-FIELD
              $A3=JUMP ADDR;MAR1=ADDR OF B(A)
              IF FALSE THEN NOT B;STEP ELSE SKIP
              IF NOT ABT THEN B=A1;SET LC1;SKIP $NO JUMP IF B(A)=C
              IFETCH-1=MPCR
              A1-1=MIR
              $DECREMENT B(A) AND JUMP
              OUTPUT1-1=CPCR
              PUTJUMP-1=MPCR
$ *****
$ ***** JUMP SY+B *****
0PR52X2:      $MIR=B-FIELD
              $JUMP TO SY+B
              $ISOLATE SY-FIELD
              $MAR=ADDR OF B-REG
              $B=(B)
              $A3=SY+B(B)
              UNCONDITIONAL JUMP LOWER *****
0PR52X3:      $A3=JUMP ADDRESS
              A3=MAR2
              A2 R=A2;CSAR;MR2;IF RDC
              20=$AR;3=LIT
              WHEN RDC THEN A2 OR 8001 L=A2,BEX
              B C=A1
              10=$AR
              A1 AND LIT=AMPCR
              B C=A1
              16=$AR
              A2 OR $MAR=A2;EXEC
              A2+1=A2;JUMP
$ *****
$ ***** JUMP OVERFLOW AND A *****
0PR53X0:      0 EOL B
              A2 C = A2;CSAR; IF FALSE THEN SET LC1
              COMP 3=$AR
              PUTJUMP-1 = AMPCR
              A2
              A2 AND 8011 C = A2; IF MST THEN SKIP
              IF NOT LC1 THEN SET LC1;JUMP ELSE JUMP
              IF LC1 THEN SET LC1;JUMP ELSE JUMP
              $JUMP IF A=C AND NO OVERFLOW OR A=1 AND OVERFLOW
              JUMP ON CD EQUAL/UNEQUAL/LIMITS *****
0PR53X1:

```


| Address | OpCode | OpCode Hex | OpCode Name | OpCode Comment | OpCode Data | OpCode Hex | OpCode Name | OpCode Comment | OpCode Data |
|---------|--------|------------|-------------|----------------|-------------|--------------------------------------|-------------|----------------|-------------|
| 07A2 | 4809 | 0C40 | 4030 | C0F0 | | B = A1 | | | |
| 07A3 | 4809 | A540 | 0C40 | C0F0 | | A1=AMPCR=AMPCR | | | |
| 07A4 | 3A00 | 0000 | 0C30 | C0C0 | | AC0-1=AMPCR | | | |
| 07A5 | 4809 | 00C0 | 0000 | C0F0 | | STEP | | | |
| 07A6 | 4824 | C0C1 | 9000 | C0F0 | | A2 C=A3,EXEC | | | |
| 07A7 | 3000 | 0000 | 000C | C030 | | 31 = SAR | | | |
| 07A8 | 4836 | E000 | 0000 | 00FC | | A3,CALL | | | |
| 07A9 | 4908 | 0F40 | 1000 | C0F0 | | BHAR = A3; SET LC1; SKIP | | | |
| 07AA | 0A50 | 0000 | 0030 | 0040 | | IFETCH-1=MPCR | | | |
| 07AB | 266C | 0000 | 0000 | 0040 | | PUTJUMP-1=MPCR | | | |
| 07AC | 4809 | 0C40 | 4030 | 80F0 | | | | | |
| 07AD | 0580 | 0000 | 0030 | 00E0 | | B=SAR,A1,LHAR | | | |
| 07AE | A809 | A15E | 003C | 08F0 | | SWITCH =LIT | | | |
| 07AF | 0040 | 00C0 | 0000 | 00E0 | | A1 GEO LIT;MR1;IF RDC | | | |
| 07B0 | 78C9 | 0000 | 0035 | 00F0 | | 4=LIT | | | |
| 07B1 | AC08 | A0C2 | 00C0 | 00F0 | | 0=CTR; IF TRUE THEN SET LC1 | | | |
| 07B2 | 68C9 | 00C0 | 00C0 | C0F0 | | WHEN RDC THEN NOT A1,BEX | | | |
| 07B3 | 4809 | 0C40 | 8800 | 00F0 | | IF ABT THEN SET LC1 | | | |
| 07B4 | 4809 | 0C40 | 0C30 | 00F0 | | B R=B | | | |
| 07B5 | 58C9 | 0000 | 0000 | 00F0 | | | | | |
| 07B6 | 2FD9 | A15E | 0000 | 00F0 | | IF LST THEN SET LC1 | | | |
| 07B7 | 0A5C | 0030 | 0030 | 0040 | | IF LC1 THEN A1 GEO LIT;SET LC1;SKIP | | | |
| 07B8 | 7801 | 0000 | 0000 | C0F0 | | IFETCH-1=MPCR | | | |
| 07B9 | 4809 | C0C1 | 4008 | 40F0 | | IF TRUE THEN WAIT | | | |
| 07BA | 810C | 0000 | 0000 | 00A0 | | A2 L=A1,LHAR,CSAR | | | |
| 07BB | 4812 | A000 | C030 | C0F0 | | SBASE=LIT;COMP 12=SAR | | | |
| 07BC | A809 | 0000 | 0030 | 08F0 | | A1 R=A1,MIR,SAVE | | | |
| 07BD | AC08 | 0000 | 0C30 | 00F0 | | MR1;IF RDC | | | |
| 07BE | 4809 | AC5E | 0030 | 00F0 | | WHEN RDC THEN BEX | | | |
| 07BF | 4429 | 0F46 | 000E | C0F0 | | A1-B=MTR,BMI | | | |
| 07C0 | 4809 | E003 | 001C | 00F0 | | IF NOT HST THEN BHAR+1=BHAR;INC;JUMP | | | |
| 07C1 | A809 | 6000 | CC00 | CCF0 | | A3=HAR2 | | | |
| 07C2 | 300C | 0000 | 0C30 | 0010 | | CTR R=A1;MR2;IF RDC | | | |
| 07C3 | AC08 | AC5D | 4C30 | 00F0 | | 11=SAR | | | |
| 07C4 | 0000 | 0000 | 0030 | 0020 | | WHEN RDC THEN A1 OR B L=A1,BEX | | | |
| 07C5 | 4809 | 0C40 | 8830 | 00F0 | | COMP 16=SAR | | | |
| 07C6 | 4809 | AC5D | 8080 | 00F0 | | R R=B | | | |
| 07C7 | 4809 | E0C0 | 1000 | 00F0 | | A1 OR B C=MIR | | | |
| 07C8 | 266C | 0000 | 0030 | 00C0 | | A3 + 1=A3 | | | |
| 07C9 | 9809 | 0000 | 0000 | 1CFC | | PUTJUMP-1=AMPCR | | | |
| 07CA | 9C28 | 0000 | 0000 | 00F0 | | MW2;IF SAI | | | |
| 07CB | 4809 | 0C40 | 4030 | 80FC | | WHEN SAI THEN 0;JUMP | | | |
| 07CC | 0580 | 0000 | 0030 | 00E0 | | | | | |
| 07CD | A809 | A15E | 0030 | 08F0 | | MANUAL JUMP | | | |
| 07CE | 0040 | 0000 | 0030 | 00E0 | | XB=A-FIELD;A3=JUMP ADDRESS | | | |
| 07CF | 78C1 | A0C2 | 0000 | 00F0 | | XSAR=A-FIELD | | | |
| 07D0 | 68C9 | 0000 | 0030 | 00F0 | | SWITCH =LIT | | | |
| 07D1 | AC08 | 00C0 | 0C30 | 00F0 | | A1 GEO LIT;MR1;IF RDC | | | |
| 07D2 | 4809 | 0C40 | 8800 | 00F0 | | 4=LIT | | | |
| 07D3 | 4809 | 0C40 | 003C | 00F0 | | NOT A1; IF TRUE THEN WAIT;SET LC1 | | | |
| 07D4 | 58C9 | 0000 | 0000 | 00F0 | | IF ABT THEN SET LC1 | | | |
| 07D5 | 28D9 | 0000 | 0000 | C0F0 | | WHEN RDC THEN BEX | | | |
| 07D6 | 0A50 | | | | | | | | |


```

080A 9000 0000 0000 0030
080B 2C19 D810 A000 00F0
080C 4809 D80F A000 00F0
080D 0BCC 0000 0000 004C

080E 227B 0000 0000 0060
080F 4809 A000 0038 00F0
0810 319C 0000 0000 0080
0811 9809 E001 0010 18F0
0812 9C08 0C46 000C 00F0
0813 4809 2F56 000C 00F0
0814 0070 0000 0030 00E0
0815 A809 0000 0000 0080
0816 AC08 0C40 0C3C 00F0
0817 A809 0C40 000C 08FC
0818 4E4B 0DC6 0820 00F0
0819 2130 0000 0000 0040
081A 4809 0C40 4001 00F0
081B A0F0 0000 0030 0080
081C 4852 0000 3B30 00F0
081D A809 A000 8B02 00F0
081E 4809 CC5C 2030 00F0
081F 4809 C012 0C00 00F0
0820 6419 CC59 0000 00F0
0821 DF80 0000 0C3C 0040
0822 7C19 0C46 0800 00FC
0823 0F90 00C0 0030 0040
0824 4809 CC5E 2030 00F0
0825 4809 E0DC 103C 00F0
0826 8419 0C40 0000 00F0
0827 0FA0 0000 0000 0040
0828 5C09 0C46 0800 00F0
0829 4809 0C41 0030 00F0
082A 8000 0000 0000 0030
082B 4809 EDC4 1030 00F0
082C 4809 A0C1 4000 00F0
082D A000 0000 0000 003C
082E 482D C001 2030 00F0
082F 3C19 E00A 1030 00F0
0830 0F80 0000 0000 004C
0831 AC08 C001 2C31 00F0
0832 982D 0C40 4000 24F0
0833 4809 E000 0C30 24FC
0834 4809 0F40 800C 00F0
0835 007C 0000 0030 0090
0836 9809 0C00 0000 18F0
0837 9C08 0F46 000C 00F0
0838 4809 2F56 000C 00F0
0839 4809 C000 0030 00F0
083A 9809 0000 0000 18F0
083B 9C08 0000 0030 00F0
083C 22A0 0000 0030 0060
083D 4809 0000 0008 00F0
083E 0190 0000 003C 00EC
083F A809 0000 0000 08F0
0840 AC08 00C0 0C0C 00F0
0841 4809 0C40 4000 00F0
0842 0BCC 0000 0000 0040

COMP 3=SAR
IF LC1 THEN A2 DR B100 C=A2,SKIP
A2 AND B011 C=A2
HALFFETCH=HPCR
***** SQUARE ROOT *****
OPR74X2:  % A3=B-FIELD/ B=A-FIELD
          PUTPAR-1=CPCR
          A1=HIR,LHAR
          CINST=LIT,COMP 8=SAR
          A3 L=BR2,MW1,IF SAI
          WHEN SAI THEN B+1=MAR
          BHAR AND LIT=MAR
          7=LIT
          MW1,IF ROC
          WHEN ROC THEN B=MAR,BEX
          B,MR1,IF ROC
          IF MR1 THEN BC01+1=B,SET LC2,STEP ELSE SKIP
          ERRORHSG-1=HPCR
          B=A1,LCTR
          15=LIT,COMP 2=SAR
          0=A3,A2,B,MR1,SET LC2,SAVE %INITIALIZE REGISTERS
          SORT1:  A1 R=B,INC
                  A2 DR B=A2,EHI
                  A2 EOL 0
                  IF FALSE THEN A2 GTR B,SKIP
                  SORT2-1=HPCR
                  IF TRUE THEN B+1=B,SKIP
                  SORT2-1=HPCR
                  A2-B=A2
                  A3 OR B001=A3
                  IF NOT C0V THEN B,SKIP
                  SORT3-1=HPCR
                  IF LST THEN B+1=B
                  B L=HIR
                  %DOUBLE B
                  COMP 1=SAR
                  A3 0AD 0=A3
                  A1 L=A1
                  COMP 2=SAR
                  A2 L=A2, JUMP
                  SORT3:  IF LC2 THEN A3 0AD 0=A3,SKIP
                          SORT4-1=HPCR
                          %A3=ANSWER,A2=RESIDUE
                          WHEN ROC THEN A2 L=A2,BEX,LCTR
                          B=A1,ASE,JUMP
                          %SELECT BR2
                          %SELECT BR?
                          %SETUP FOR A(B)
                          BHAR R=MAR
                          8=SAR,7=LIT
                          MW1,IF SAI
                          WHEN SAI THEN BHAR+1=MAR
                          BHAR AND LIT=MAR
                          A2 = HIR
                          MW1,IF SAI
                          WHEN SAI THEN 0,STEP
                          GETPAR-1=CPCR
                          LHAR
                          CINST=LIT
                          MR1,IF ROC
                          WHEN ROC THEN BEX
                          B=A1
                          HALFFETCH=HPCR
                          ***** LOAD B(A) WITH BCB *****
                          %RESTORE A1 TO CURRENT INSTRUCTION
                          %ENTER HALFFETCH AT +1
                          ***** 24450C00 D

23860000 D
23870C00 D
23880C00 D
23890C00 D
23900C00 D
23910C00 D
23920C00 D
23930C00 D
23940C00 D
23950C00 D
23960C00 D
23970C00 D
23980C00 D
23990C00 D
24000C00 D
24010C00 D
24020C00 D
24030C00 D
24040C00 D
24050C00 D
24060C00 D
24070C00 D
24080C00 D
24090C00 D
24100C00 D
24110C00 D
24120C00 D
24130C00 D
24140C00 D
24150C00 D
24160C00 D
24170C00 D
24180C00 D
24190C00 D
24200C00 D
24210C00 D
24220C00 D
24230C00 C
24240C00 D
24250C00 D
24260C00 D
24270C00 D
24280C00 D
24290C00 D
24300C00 D
24310000 D
24320C00 D
24330C00 D
24340C00 D
24350C00 D
24360C00 D
24370C00 D
24380C00 D
24390C00 D
24400C00 D
24410C00 D
24420C00 D
24430C00 D
24440000 D
24450C00 D

```


| | | | | | | | | | |
|------|------|------|------|-------|----------|--|------------------------------------|----------|---|
| 0843 | 4809 | E140 | 003C | 00F0 | OPR74X3: | %MIR=A-FIELD; A3=B-FIELD | | 24460C00 | 0 |
| 0844 | 0080 | 0000 | 0030 | 00E0 | | A3=LIT=HAR | %HAR=B(B) | 24470C00 | 0 |
| 0845 | A809 | 0000 | 0000 | 008C | | BBASE=LIT | | 24480C00 | 0 |
| 0846 | AC08 | 0000 | 0C00 | 00F0 | | MR1,IF RDC | | 24490C00 | 0 |
| 0847 | 4809 | 0C40 | 0030 | 00F0 | | WHEN RDC THEN BEX | %READ B(B) | 24500C00 | 0 |
| 0848 | 4809 | 0C52 | 0000 | 00F0 | | B=MIR,BMI | | 24510C00 | 0 |
| 0849 | 6408 | 2C40 | 000C | 00F0 | | 0 EOL B | %CHECK FOR B(0) => ILLEGAL | 24520C00 | 0 |
| 084A | 08B0 | 0003 | 0000 | C040 | | IF FALSE THEN LIT + B=HAR,STEP ELSE SKIP | | 24530C00 | 0 |
| 084B | 08C0 | 0000 | 0000 | 0040 | | HALFFETCH-1=HPCR | %HAR=B(A) ADDRESS | 24540C00 | 0 |
| 084C | 08D0 | 0000 | 0000 | 0040 | | HALFFETCH=HPCR | %ENTER AT +1 AND DO NOT WRITE B(0) | 24550C00 | 0 |
| 084D | 4809 | E000 | 003C | 00F0 | | %***** COMPARE REGISTERS ***** | | 24560C00 | 0 |
| 084E | AC08 | 0000 | 0C00 | 00F0 | | OPR74X4: %A3=A(B); B=A(A) | | 24570C00 | 0 |
| 084F | 4809 | 0C40 | 1000 | 00F0 | | A3=HAR | %READ A(B) | 24580C00 | 0 |
| 0850 | 4809 | 0C40 | 003C | 00F0 | | MR1,IF RDC | | 24590C00 | 0 |
| 0851 | A809 | 00C0 | 0000 | 008F0 | | WHEN RDC THEN BEX | %A3=(A(B)) | 24600C00 | 0 |
| 0852 | AC08 | 0000 | 0C00 | 00F0 | | B=A3,BMI | %HAR = ADDRESS OF A(A) | 24610C00 | 0 |
| 0853 | 2860 | 00C0 | 0030 | 0040 | | B=HAR | | 24620C00 | 0 |
| 0854 | 4809 | 0C40 | 003C | 20F0 | | MR1,IF RDC | %A3=A(B);B=A(A) | 24630C00 | 0 |
| 0855 | 08C0 | 0000 | 0C30 | 00C0 | | WHEN RDC THEN BEX | | 24640C00 | 0 |
| 0856 | A809 | 0040 | 0000 | 008F0 | | SETCOMPARE-1=HPCR | | 24650C00 | 0 |
| 0857 | AC08 | 0F46 | 0C0C | C0F0 | | %***** COMPARE A(B) WITH A(A+1); A(A) SET C***** | | 24660C00 | 0 |
| 0858 | 4809 | 2F56 | 000C | 00F0 | | OPR74X5: %B,MIR=A-FIELD; A3=B-FIELD | | 24670000 | 0 |
| 0859 | A809 | 0C40 | 0030 | 008F0 | | B=HAR,ASR | %SETUP FOR BHAR1 | 24680C00 | 0 |
| 085A | AC08 | E0C0 | 0C0C | 00F0 | | HALFFETCH=HPCR | | 24690C00 | 0 |
| 085B | A809 | 0C40 | 1030 | 008F0 | | MR1,IF RDC | | 24700C00 | 0 |
| 085C | AC08 | 00C0 | 0C30 | 00F0 | | WHEN RDC THEN BHAR+1=HAR,BEX | | 24710C00 | 0 |
| 085D | 4809 | C0C1 | A000 | 40F0 | | LIT AND BHAR=HAR | %CONVERT A(B) TO A(0) | 24720C00 | 0 |
| 085E | A000 | 0000 | 0030 | 0030 | | B=MIR,MR1,IF RDC | %MIR=(A(A)) | 24730C00 | 0 |
| 085F | 4809 | EC4C | 0C00 | 00F0 | | WHEN RDC THEN A3=HAR,BEX | %B=(A(A+1)) | 24740C00 | 0 |
| 0860 | 4808 | EC58 | 0000 | C0F0 | | B=A3,MR1,IF RDC | %A3=(A(A+1)) | 24750C00 | 0 |
| 0861 | 7C2B | D81D | A000 | C0F0 | | WHEN RDC THEN BEX | %B=(A(B)) | 24760C00 | 0 |
| 0862 | 7429 | D81D | A000 | C0F0 | | A2 C=A2,CSAR | %LIMITS BIT IS MSB | 24770C00 | 0 |
| 0863 | 4809 | 0C40 | 1030 | 00F0 | | COMP 2=5AR | | 24780000 | 0 |
| 0864 | 4809 | EC4C | 0000 | 00F0 | | A3 XOR B | %TEST DIFFERENT SIGNS | 24790C00 | 0 |
| 0865 | 4808 | EC5E | 0000 | 00F0 | | A3 LEO B; IF HST THEN STEP ELSE SKIP | | 24800C00 | 0 |
| 0866 | 7C2B | D81D | A000 | 00F0 | | IF FALSE THEN A2 OR B100 C=A2;JUMP ELSE SKIP | | 24810C00 | 0 |
| 0867 | 7429 | D81D | A030 | C0F0 | | IF TRUE THEN A2 OR B100 C=A2;JUMP %SAME SIGNS | | 24820C00 | C |
| 0868 | 482D | D80F | A000 | C0F0 | | B=A3,BMI | | 24830C00 | 0 |
| 0869 | 4809 | 0C40 | 0030 | 00F0 | | A3 XOR B | %A3=A(B);B=A(A) | 24840C00 | 0 |
| 086A | A809 | 0000 | 0000 | 008F0 | | A3 LSS B; IF HST THEN STEP ELSE SKIP | | 24850000 | 0 |
| 086B | AC08 | 0F46 | 0C3C | C0F0 | | IF FALSE THEN A2 OR B100 C=A2;JUMP ELSE SKIP | | 24860C00 | 0 |
| 086C | 4809 | 2F56 | 000C | 00F0 | | IF TRUE THEN A2 OR B100 C=A2;JUMP | | 24870C00 | 0 |
| 086D | A809 | 0C40 | 0030 | C0F0 | | A2 AND B011 C=A2;JUMP | | 24880C00 | 0 |
| 086E | AC08 | E000 | 0C0 | | | | | | |

0393 3900 0000 0000 0000 0000
0392 3980 0000 0000 0000 0000
0391 3990 0000 0000 0000 0000
0390 3970 0000 0000 0000 0000
0389 38F0 0000 0000 0000 0000
0380 3C10 0000 0000 0000 0000
0377 37D0 0000 0000 0000 0000
0366 3720 0000 0000 0000 0000
035C 3610 0000 0000 0000 0000
0349 3540 0000 0000 0000 0000
02E5 3860 0000 0000 0000 0000
02E3 3A50 0000 0000 0000 0000
02CF 2030 0000 0000 0000 0000
02C6 2C80 0000 0000 0000 0000
028C 2C30 0000 0000 0000 0000
0286 28AC 0000 0000 0000 0000
022A 2300 0000 0000 0000 0000
01FC 2090 0000 0000 0000 0000
01F8 2090 0000 0000 0000 0000
01F4 20A0 0000 0000 0000 0000
01F0 20A0 0000 0000 0000 0000
01EE 2320 0000 0000 0000 0000
01E7 2320 0000 0000 0000 0000
010F 2320 0000 0000 0000 0000
0107 2320 0000 0000 0000 0000
01CF 2320 0000 0000 0000 0000
01CD 2320 0000 0000 0000 0000
01C6 2320 0000 0000 0000 0000
018E 1E70 0000 0000 0000 0000
018D 10F0 0000 0000 0000 0000
018C 1070 0000 0000 0000 0000
0188 1CF0 0000 0000 0000 0000
018A 1CDC 0000 0000 0000 0000
0189 1C60 0000 0000 0000 0000
0188 18E0 0000 0000 0000 0000
0187 1970 0000 0000 0000 0000
0176 17E0 0000 0000 0000 0000
0175 1770 0000 0000 0000 0000
0172 17A0 0000 0000 0000 0000
0161 16C0 0000 0000 0000 0000
015E 1860 0000 0000 0000 0000
0144 1440 0000 0000 0000 0000
0129 1370 0000 0000 0000 0000
0126 1440 0000 0000 0000 0000
0CF8 1150 0000 0000 0000 0000
0CFA 1110 0000 0000 0000 0000
00F9 1000 0000 0000 0000 0000
0CF8 10A0 0000 0000 0000 0000
0CF7 1080 0000 0000 0000 0000
0CF6 1020 0000 0000 0000 0000
00F5 0F80 0000 0000 0000 0000
0CED 0F30 0000 0000 0000 0000
0CD8 0DF0 0000 0000 0000 0000
0FD4 1190 0000 0000 0000 0000
00CA 1EEN 0000 0000 0000 0000
0CC8 2270 0000 0000 0000 0000
0CB3 3370 0000 0000 0000 0000
00AA 2CAC 0000 0000 0000 0000
0CAB 1EE0 0000 0000 0000 0000
CCA6 2270 0000 0000 0000 0000

0CA2 22A0 0000 0000 0000 0060
0C9F 21E0 0000 0C00 0040
0C9E 8700 C0C0 0000 00C0
0C9D 87C0 0000 0C30 00C0
0C9C 8780 00C0 0000 00C0
0C9B 21E0 0000 0C00 C040
0C9A 21E0 00C0 0030 0040
0C99 87A0 0000 0C00 00C0
0C98 879C 00C0 0030 00C0
0C97 871C 0000 0000 00C0
0C96 868C 0000 0000 C0C0
0C95 8530 00C0 0030 C0C0
0C94 8480 0000 0030 C0C0
0C93 8420 0000 0030 C0C0
0C92 8000 00C0 003C 00C0
0C91 7F30 0000 0030 C0C0
0C90 7E30 0000 0000 00C0
0C8F 21E0 00C0 0030 0040
0C8E 21E0 0000 0030 0040
0C8D 21E0 0000 0030 0040
0C8C 21E0 0000 0C30 0040
0C8B 7080 0000 0C30 C0CC
0C8A 7090 0000 0000 00CC
0C89 7080 00C0 003C 00C0
0C88 7070 0000 0C00 00C0
0C87 2100 0000 0030 00C0
0C86 2100 0000 0000 00C0
0C85 2100 0000 0000 00C0
0C84 7CA0 0000 0030 00C0
0C83 7AB0 0000 0C30 00C0
0C82 7A10 0000 0030 00C0
0C81 7990 0000 0030 00C0
0C80 2100 0000 0030 C0C0
0C7F 2100 0000 003C 00C0
0C7E 2100 0000 0000 00C0
0C7D 2100 0000 0030 C0CC
0C7C 78EC 0000 0000 00C0
0C7B 7870 00C0 0030 00C0
0C7A 7800 0000 0000 00C0
0C79 7780 C000 0000 00C0
0C78 210C 00C0 0030 00C0
0C77 2100 0000 0C00 00C0
0C76 2100 0000 0000 00C0
0C75 2100 00C0 0030 00C0
0C74 7780 0000 0C00 C0C0
0C73 7750 0000 003C 00C0
0C72 7730 0000 0000 00C0
0C71 7700 0000 0C00 C0C0
0C70 210C 0000 0000 00C0
0C6F 210C 0000 0030 00CC
0C6E 210C 0000 0030 00C0
0C6D 210C 0000 0030 00C0
0C6C 76E0 0000 0000 00C0
0C6B 7630 0000 0000 00C0
0C6A 7610 0000 0030 00C0
0C69 7560 0000 0C00 00C0
0C68 2100 0000 0000 00C0
0C67 73AC 0000 0C30 00C0
0C66 7390 0000 0C30 00C0
0C65 7320 0000 0000 C0C0

| | | | | |
|------|------|------|------|------|
| 0c64 | 7310 | 0000 | 0c30 | 00c0 |
| 0c63 | 730c | 00c0 | c030 | c0c0 |
| 0c62 | 72f0 | c0c0 | 0000 | 00c0 |
| 0c61 | 72e0 | 0000 | 0c30 | c0c0 |
| 0c60 | 72c0 | 0000 | 0000 | 00c0 |
| 0c5f | 72a0 | 0000 | 0030 | 00c0 |
| 0c5e | 7280 | 0000 | 0000 | c0c0 |
| 0c5d | 7260 | 0000 | 0020 | 00c0 |
| 0c5c | 7240 | 0000 | 0c00 | c0c0 |
| 0c5b | 7220 | 0000 | 0020 | 00c0 |
| 0c5a | 7200 | 00c0 | 0c00 | 00c0 |
| 0c59 | 71ec | 00c0 | 0000 | 00c0 |
| 0c58 | 7100 | 0000 | 0000 | c0c0 |
| 0c57 | 7100 | 00c0 | 0030 | 00c0 |
| 0c56 | 7100 | 00c0 | 0020 | 00c0 |
| 0c55 | 7100 | 0000 | 0c00 | 00c0 |
| 0c54 | 701c | 0000 | 0030 | c0cc |
| 0c53 | 5ed0 | 0000 | 0c30 | 00c0 |
| 0c52 | 5d70 | 0000 | 0c30 | 00c0 |
| 0c51 | 5cfc | 00c0 | 0c00 | 00c0 |
| 0c50 | 68fc | 0000 | 0020 | 00c0 |
| 0c4f | 6830 | 0000 | 0020 | 00c0 |
| 0c4e | 6a80 | 0000 | 0c30 | 00c0 |
| 0c4d | 6a10 | 00c0 | 0020 | c0c0 |
| 0c4c | 6990 | 0000 | 0c30 | c0c0 |
| 0c4b | 6970 | 0000 | 0c00 | c0c0 |
| 0c4a | 2100 | 0000 | 0020 | 00c0 |
| 0c49 | 6940 | 0000 | 0030 | 00c0 |
| 0c48 | 6900 | 0000 | 0000 | 00cc |
| 0c47 | 6880 | 00c0 | 0020 | 00cc |
| 0c46 | 6860 | 00c0 | 0030 | 00c0 |
| 0c45 | 67e0 | 0000 | 0020 | 00c0 |
| 0c44 | 67c0 | 0000 | 0020 | 00cc |
| 0c43 | 6740 | 0000 | 0c00 | 00c0 |
| 0c42 | 672c | 0000 | 0020 | c0c0 |
| 0c41 | 6700 | 0000 | 0020 | 00c0 |
| 0c40 | 66ac | 0000 | 0000 | 00c0 |
| 0c3f | 21ec | 0000 | 0000 | c040 |
| 0c3e | 21ec | 0000 | 0000 | 0040 |
| 0c3d | 6610 | 0000 | 0c00 | c0c0 |
| 0c3c | 21ec | 0000 | 0020 | 0040 |
| 0c3b | 21e0 | 0000 | 0020 | 0040 |
| 0c3a | 647c | 0000 | 0000 | 00c0 |
| 0c39 | 640c | 0c00 | 0020 | c0c0 |
| 0c38 | 6240 | 0c00 | 0000 | 00c0 |
| 0c37 | 61f0 | 00c0 | 0c00 | 00c0 |
| 0c36 | 60a0 | 0000 | 0c00 | 00c0 |
| 0c35 | 6070 | 0000 | 0020 | 00c0 |
| 0c34 | 5ef0 | 0000 | 0000 | 00c0 |
| 0c33 | 5e80 | 00c0 | 0020 | 00c0 |
| 0c32 | 5d00 | 0000 | 0000 | 00c0 |
| 0c31 | 5c00 | c000 | 0030 | 00c0 |
| 0c30 | 5c90 | 0000 | 0020 | 00c0 |
| 0c2f | 58c0 | 00c0 | 0020 | 00c0 |
| 0c2e | 5880 | 00c0 | 0000 | 00c0 |
| 0c2d | 5a00 | 0000 | 0000 | c0c0 |
| 0c2c | 5a50 | 0000 | 0020 | 00c0 |
| 0c2b | 5990 | 0000 | 0000 | 00c0 |
| 0c2a | 58f0 | 0000 | 0020 | c0c0 |
| 0c29 | 584c | 0000 | 0c00 | 00c0 |

| | | | | | |
|------|------|------|------|------|------|
| 0028 | 5790 | 0000 | 0000 | 0000 | 0000 |
| 0027 | 5700 | 0000 | 0000 | 0000 | 0000 |
| 0026 | 5650 | 0000 | 0000 | 0000 | 0000 |
| 0025 | 5550 | 0000 | 0000 | 0000 | 0000 |
| 0024 | 5480 | 0000 | 0000 | 0000 | 0000 |
| 0023 | 5410 | 0000 | 0000 | 0000 | 0000 |
| 0022 | 5270 | 0000 | 0000 | 0000 | 0000 |
| 0021 | 5190 | 0000 | 0000 | 0000 | 0000 |
| 0020 | 5130 | 0000 | 0000 | 0000 | 0000 |
| 001F | 5050 | 0000 | 0000 | 0000 | 0000 |
| 001E | 5070 | 0000 | 0000 | 0000 | 0000 |
| 001D | 4F00 | 0000 | 0000 | 0000 | 0000 |
| 001C | 4F80 | 0000 | 0000 | 0000 | 0000 |
| 001B | 4F10 | 0000 | 0000 | 0000 | 0000 |
| 001A | 2100 | 0000 | 0000 | 0000 | 0000 |
| 0019 | 2100 | 0000 | 0000 | 0000 | 0000 |
| 0018 | 4EA0 | 0000 | 0000 | 0000 | 0000 |
| 0017 | 4E50 | 0000 | 0000 | 0000 | 0000 |
| 0016 | 4CF0 | 0000 | 0000 | 0000 | 0000 |
| 0015 | 4CA0 | 0000 | 0000 | 0000 | 0000 |
| 0014 | 4B00 | 0000 | 0000 | 0000 | 0000 |
| 0013 | 4B00 | 0000 | 0000 | 0000 | 0000 |
| 0012 | 4A30 | 0000 | 0000 | 0000 | 0000 |
| 0011 | 4980 | 0000 | 0000 | 0000 | 0000 |
| 0010 | 4960 | 0000 | 0000 | 0000 | 0000 |
| 000F | 4920 | 0000 | 0000 | 0000 | 0000 |
| 000E | 4870 | 0000 | 0000 | 0000 | 0000 |
| 000D | 47F0 | 0000 | 0000 | 0000 | 0000 |
| 000C | 4770 | 0000 | 0000 | 0000 | 0000 |
| 000B | 4680 | 0000 | 0000 | 0000 | 0000 |
| 000A | 45F0 | 0000 | 0000 | 0000 | 0000 |
| 0009 | 45C0 | 0000 | 0000 | 0000 | 0000 |
| 0008 | 4530 | 0000 | 0000 | 0000 | 0000 |
| 0007 | 4520 | 0000 | 0000 | 0000 | 0000 |
| 0006 | 44A0 | 0000 | 0000 | 0000 | 0000 |
| 0005 | 2100 | 0000 | 0000 | 0000 | 0000 |
| 0004 | 4400 | 0000 | 0000 | 0000 | 0000 |
| 0003 | 4360 | 0000 | 0000 | 0000 | 0000 |
| 0002 | 42C0 | 0000 | 0000 | 0000 | 0000 |
| 0001 | 2100 | 0000 | 0000 | 0000 | 0000 |
| 0000 | 09F0 | 0000 | 0000 | 0000 | 0000 |

0 ERRORS. 2536 CARDS. 14930 TOKENS. 26338 RULES. 1792 SECONDS.
 0 WARNINGS. 100 DISK SECTORS.

APPENDIX C. LOADER PROGRAM LISTING

This appendix provides a copy of the Microtranslator output listing of the Loader written in conjunction with the Emulator. Its source is maintained on disk and cards; and its object module is maintained on disk. The Loader actually consists of three separate programs which provide assembly, debugging and IOC functions to the Emulator. It is a by-product of the Emulation, and it was used for implementation and testing of the Emulator. Its functions should be incorporated in the Emulator Program in the course of expansion to a full emulation.

\$MERGE LOADER-SOURCE
PROGRAM UYK7-LOADER

```

ABASE=0
BBASE=8
SBASE=16
PAR=29
SWITCH=88
PLOC=120
HEXADDR=210
CARDBUF=212
COL5X8=213
COL9X12=214
COL13X16=215
LASTCOL=232
PRINTADD=237
BLANKLINE=500
ERRORLIST=280. PRINTAREA=238. DISKADDR=3132
NUMLINES=36
*
* BEGIN PROGRAM HERE BY ZEROING MEMORY
WAIT
START: %RETURN HERE ON SIGNAL FROM EVALUATOR TO RESTART
      SET GC1 WHEN GC1 THEN 1 L=A1,MAR2
      COMP 16 =: SAR
      O=1 MIR; SAVE
      MW2: A1-1 =: A1; IF SAI
      IF AGT THEN SKIP ELSE STEP
      WHEN SAI THEN A1=: MAR2; JUMP
*
* READ CARDS IN TO SETUP ERROR ROUTINES
ERRORSETUP:
      B110=MAR2
      LIT L=A1
      MOST(DISKADDR)=LIT; COMP 8=SAR
      A1 OR LIT=A1
      LEAST(DISKADDR)=LIT
      LIT L=B
      COMP 16=SAR; 180=LIT
      A1 OR B=MIR
      OUTPUT1-1=CPCR
      AMPCR=A1
      ERRORLIST=AMPCR
      LIT L=B
      3=LIT;COMP 16=SAR
      A1 OR B = MIR
ERRORSETUP1:
      EXTOP-1=CPCCR
      ERRORSETUP1-1=MPCCR
*
* SETUP BUFFER OF ALL SPACES FOR PRINTING BLANK LINES
O=BC8,LC1R
32=LIT
BLANKLINE=AMPCR
B=MIR
AMPCR=MAR2
SAVE
MW2: IF SAI

```


| | | | | | | | |
|------|------|------|------|------|--|----------|---|
| 004C | 0040 | 00C0 | 0000 | C0E0 | CARDBUF=LIT | 0118C000 | D |
| 004D | 0100 | 0000 | 0030 | 0060 | READAFIELD-1=:CPCR | 0119C000 | D |
| 004E | 4809 | 2000 | 4000 | C0F0 | LIT=:A1 | 0120C000 | D |
| 004F | 0070 | 00C0 | 0030 | 00A0 | 7=LIT;COMP 16=SAR | 0121C000 | D |
| 0050 | 4812 | 0041 | 0800 | C0F0 | B L=:B; SAVE | 0122C000 | D |
| | | | | | %SETUP FOR TWICE TRHU LOOP | 0123C000 | D |
| | | | | | %LOOP BEGINS HERE | 0124C000 | D |
| | | | | | % | 0125C000 | D |
| 0051 | 4809 | E001 | 1030 | C0F0 | A3 L=:A3 | 0126C000 | D |
| 0052 | 9000 | 0000 | 0030 | 0030 | COMP 3=:SAR | 0127C000 | D |
| 0053 | 4809 | 0C41 | 8B90 | 00FC | B C=:B;MIR | 0128C000 | D |
| 0054 | 0000 | 00C0 | 0000 | 0030 | COMP 8=:SAR | 0129C000 | D |
| 0055 | 4809 | AC56 | 0800 | 00F0 | A1 AND B=:B | 0130C000 | D |
| 0056 | 4836 | E05C | 1030 | 00F0 | B OR A3=:A3;BHI; CALL | 0131C000 | D |
| | | | | | %END OF LOOP | 0132C000 | D |
| | | | | | % | 0133C000 | D |
| | | | | | %OPCODE IS NOW IN LSB OF B | 0134C000 | D |
| | | | | | %IF NOT ADV THEN OPC<60 =>TYPE 1 | 0135C000 | D |
| | | | | | A3 LSS LIT | 0136C000 | D |
| | | | | | 4B=:LIT | 0137C000 | D |
| | | | | | %OPCODE = 60 OCTAL | 0138C000 | D |
| | | | | | IF FALSE THEN A3 LEO LIT;SKIP | 0139C000 | D |
| | | | | | TYPE1-1=:MPCR | 0140C000 | D |
| | | | | | %IF NOT ADV THEN OPC<=61 =>TYPE 4A | 0141C000 | D |
| | | | | | 4B=:LIT | 0142C000 | D |
| | | | | | IF FALSE THEN A3 GEO LIT;SKIP | 0143C000 | D |
| | | | | | TYPE4A-1=:MPCR | 0144C000 | D |
| | | | | | %IF ADV THEN OPCODE>=70 =>TYPE 4A | 0145C000 | D |
| | | | | | %ELSE TYPE 4D | 0146C000 | D |
| | | | | | 5C=:LIT | 0147C000 | D |
| | | | | | IF FALSE THEN SKIP | 0148C000 | D |
| | | | | | TYPE4A-1=:MPCR | 0149C000 | D |
| | | | | | %INSTRUCTION IS TYPE 4B - OPCODE STILL IN A3 | 0150C000 | D |
| | | | | | TYPE4B: | 0151C000 | D |
| | | | | | COL5X8=LIT | 0152C000 | D |
| | | | | | READAFIELD-1=:CPCR | 0153C000 | D |
| | | | | | A3 L=:A3 | 0154C000 | C |
| | | | | | COMP 3=:SAR | 0155C000 | D |
| | | | | | B C=:B;MIR | 0156C000 | D |
| | | | | | COMP 8=:SAR | 0157C000 | D |
| | | | | | A1 AND B=:B | 0158C000 | D |
| | | | | | A3 OR B=:A3;BHI | 0159C000 | D |
| | | | | | B C=:B;MIR | 0160C000 | D |
| | | | | | A3 L=:A3 | 0161C000 | D |
| | | | | | COMP 1=:SAR | 0162C000 | D |
| | | | | | A3 OR 800T=:A3;BHI;SAVE | 0163C000 | D |
| | | | | | %SETUP FOR TWO PASS LOOP FOR REST OF "H" | 0164C000 | D |
| | | | | | A3 L=:A3 | 0165C000 | D |
| | | | | | COMP 3=:SAR | 0166C000 | D |
| | | | | | B C=:B;MIR | 0167C000 | D |
| | | | | | COMP 8=:SAR | 0168C000 | D |
| | | | | | A1 AND B=:B | 0169C000 | D |
| | | | | | A3 OR B=:A3;BHI;CALL | 0170C000 | D |
| | | | | | %END OF LOOP | 0171C000 | D |
| | | | | | OUTPUT-1=:MPCR | 0172C000 | D |
| | | | | | % | 0173C000 | D |
| | | | | | %WORD WAS PULIT INTO LOWER A3 | 0174C000 | D |
| | | | | | TYPE4A: | 0175C000 | D |
| | | | | | %OPCODE IS STILL IN A3 | 0176C000 | D |
| | | | | | COL5X8=LIT | 0177C000 | D |
| | | | | | READAFIELD-1=:CPCR | | |
| | | | | | SET LC2;SAVE | | |
| | | | | | %START 3 PASS LOOP HERE | | |
| | | | | | A3 L=:A3 | | |

| | | | | | | | |
|------|------|------|------|------|--|--|--|
| 0057 | 4809 | E15E | 0000 | 00F0 | | | |
| 0058 | 0300 | 0000 | 0030 | 00E0 | | | |
| 0059 | 7C19 | E158 | 0000 | 00F0 | | | |
| 005A | 0110 | 0000 | 0030 | C040 | | | |
| 005B | 0310 | 0000 | 0000 | 00E0 | | | |
| 005C | 7C19 | E15E | 0000 | 00F0 | | | |
| 005D | 0120 | 0000 | 0000 | 0040 | | | |
| 005E | 0380 | 0000 | 0030 | 00E0 | | | |
| 005F | 7019 | 0000 | 0030 | 00F0 | | | |
| 0060 | 0130 | 0000 | 0000 | 0040 | | | |
| 0061 | 0D50 | 00C0 | 0030 | 00E0 | | | |
| 0062 | 0140 | 0000 | 0000 | 0060 | | | |
| 0063 | 4809 | E001 | 1000 | C0F0 | | | |
| 0064 | 9000 | 00C0 | 0030 | 0030 | | | |
| 0065 | 4809 | 0C41 | 8B90 | 00F0 | | | |
| 0066 | 0000 | 0000 | 0000 | 0030 | | | |
| 0067 | 4809 | AC56 | 0800 | 00F0 | | | |
| 0068 | 4809 | EC5C | 1000 | 00F0 | | | |
| 0069 | 4809 | 0C41 | 8B90 | C0F0 | | | |
| 006A | 4809 | E001 | 1030 | C0FC | | | |
| 006B | 8000 | 0000 | 0000 | C030 | | | |
| 006C | 4812 | E05C | 1000 | 00F0 | | | |
| 006D | 4809 | E301 | 1030 | 00F0 | | | |
| 006E | 9000 | 00C0 | 0030 | C030 | | | |
| 006F | 4809 | 0C41 | 8B90 | 00F0 | | | |
| 0070 | 0000 | 0000 | 0000 | C030 | | | |
| 0071 | 4809 | AC56 | 0800 | C0F0 | | | |
| 0072 | 4836 | EC5C | 1030 | 00F0 | | | |
| 0073 | 0150 | 0000 | 0030 | 0040 | | | |
| 0074 | 0D50 | 00C0 | 0030 | 00E0 | | | |
| 0075 | 0160 | 0000 | 0000 | 0060 | | | |
| 0076 | 4852 | 0000 | 0000 | 00F0 | | | |
| 0077 | 4809 | E001 | 1000 | 00F0 | | | |


```

0078 9000 0000 0030 0030
0079 4809 0C41 8880 00FC
007A 0000 0000 0000 0030
007B 4809 AC56 0800 00F0
007C 4809 EC5C 1030 00F0
007D 382E 0000 0030 00F0

007E 4809 E0C1 1C30 00FC
007F 900C 0000 0000 0030
0080 4809 E05C 1030 00FC
0081 0170 0000 0000 0040

0082 0050 0000 0030 00E0
0083 0180 0000 0000 0060
0084 4852 0000 0030 00F0
0085 4809 E001 1C30 00F0
0086 9000 0000 0030 0030
0087 4809 0C41 8880 00F0
0088 0000 0000 0030 0030
0089 4809 AC56 0800 00F0
008A 4809 EC5C 1030 00F0
008B 382E 0000 0000 00F0

008C 4809 E001 1C30 00F0
008D 900C 0000 0000 0030
008E 4809 E05C 1000 00F0
008F 0060 0000 0030 00E0
0090 0190 0000 0030 0060
0091 4809 E0F1 1C00 00F0
0092 9000 0000 0000 0030
0093 4809 0C41 8880 00F0
0094 0000 0000 0030 0030
0095 4809 AC56 0800 00F0
0096 4809 EC5D 1030 00F0
0097 8000 0000 0030 0030
0098 4809 0C41 8880 00F0
0099 9000 0000 0030 0030

009A 4849 E05C 1000 00FC
009B 4812 0000 0030 00F0
009C 4809 E001 1030 00F0
009D 9000 0000 0000 0030
009E 4809 0C41 8880 00F0
009F 0000 0000 0030 0030
00A0 4809 AC56 0800 00F0
00A1 4836 EC5C 1030 00F0

00A2 0070 0000 0000 00E0
00A3 01A0 0000 0030 0060
00A4 3419 E000 0030 00F0
00A5 09AC 0000 0000 0040

```

```

      COMP 3=:SAR
      B C=:B,MIR
      COMP 8=:SAR
      A1 AND B=:B
      A3 OR B=:A3,BMI
      IF LC2 THEN JUMP ELSE CALL
      XEVD LOOP
      *
      A3 L=:A3,BEX
      COMP 1=:SAR
      *INSERT 1 FIELD
      A3 OR B001=:A3
      OUTPUT-1=:MPCR
      *RESULTS IN A3
      *
      TYPE1:      *OPCODE IS IN A3
      COL5X8=LIT
      READAFIELD-1=:CPCR
      SET LC2,SAVE
      *START 3 PASS LOOP HERE
      A3 L=:A3
      COMP 3=:SAR
      B C=:B,MIR
      COMP 8=:SAR
      A1 AND B=:B
      A3 OR B=:A3,BMI
      IF LC2 THEN JUMP ELSE CALL
      XEVD LOOP A,K,B FIELDS INSERTED NOW GET 1 FIELD
      A3 L=:A3,BEX
      COMP 1=:SAR
      A3 OR B001=:A3
      COL9X12=LIT
      READAFIELD-1=:CPCR
      A3 L=:A3
      COMP 3=:SAR
      B C=:B,MIR
      COMP 8=:SAR
      A1 AND B=:B
      A3 OR B L=:A3,BMI
      COMP 1=:SAR
      B C=:B,MIR
      COMP 8=:SAR
      *MSB OF Y HAS BEEN INSERTED
      A3 OR B001=:A3,BMI,SET LC2
      *START TWO - TWO PASS LOOPS HERE FOR REMAINDER OF Y FIELD
      YFIELD: SAVE
      A3 L=:A3
      COMP 3=:SAR
      B C=:B,MIR
      COMP 8=:SAR
      A1 AND B=:B
      A3 OR B=:A3,BMI, CALL
      XEVD OF LOOP
      *
      COL13X16=LIT
      READAFIELD-1=:CPCR
      IF NOT LC2 THEN A3=MIR, SKIP ELSE STEP
      YFIELD-1=:MPCR
      *COMPLETED Y
      *COMPLETED TYPE 1 NOW SAVE IN MIR AND OUTPUT

```

```

01780000 D
01790000 D
01800000 D
01810000 D
01820000 D
01830000 D
01840000 D
01850000 D
01860000 D
01870000 D
01880000 D
01890000 D
01900000 D
01910000 D
01920000 D
01930000 D
01940000 D
01950000 D
01960000 D
01970000 D
01980000 D
01990000 D
02000000 D
02010000 D
02020000 D
02030000 D
02040000 D
02050000 D
02060000 D
02070000 D
02080000 D
02090000 D
02100000 D
02110000 D
02120000 D
02130000 D
02140000 D
02150000 D
02160000 D
02170000 D
02180000 D
02190000 D
02200000 D
02210000 D
02220000 D
02230000 D
02240000 D
02250000 D
02260000 D
02270000 D
02280000 D
02290000 D
02300000 D
02310000 D
02320000 D
02330000 D
02340000 D
02350000 D
02360000 D
02370000 D

```



```

00C5 0240 0000 0000 0040
00C6 0250 0000 0000 0060
00C7 4809 E000 0000 00FC
00C8 0580 0000 0000 00E0
00C9 4809 2000 0000 00F0
00CA 0260 0000 0000 0040

00C8 0270 0000 0000 0060
00C9 4809 E000 0000 00FC
00CA 4809 2000 0000 00F0
00CB 0100 0000 0000 00E0
00CC 0280 0000 0000 0060
00CD 4949 0000 0000 00FC
00CE 0290 0000 0000 0040

00D2 02A0 0000 0000 0060
00D3 4809 00C1 0800 00FC
00D4 A100 0000 0000 00A0
00D5 4809 EC40 1000 00F0
00D6 4809 2000 0000 00F0
00D7 4809 00C1 4000 00FC
00D8 3070 0000 0000 00A0
00D9 4812 0000 0000 00FC
00DA 9809 0000 0000 1CFC
00DB 9C08 AC40 0C32 00FC
00DC 8429 0F45 0C1C 00FC
00DD 4809 E000 0800 00FC
00DE 4809 2000 0000 00FC
00DF 0780 0000 0000 00E0

00E0 4809 C0C0 A000 40FC
00E1 0000 0000 0000 0020
00E2 4809 C001 2000 00FC
00E3 4809 CC5C 2000 00FC
00E4 9809 0000 0000 1CFC
00E5 9C08 0000 0000 00FC

SETA00R-1=:MPCR
XEND SETAREG
X

SETSWITCHES:
X SWITCH VALUE WILL BE IN COL 4 OF CARD OR BITS 12-14 OF A3
X MASK VALUE AND WRITE TO ADDRESS 64
GETA00R-1=:CPCR
A3=:MIR
SWITCH=:LIT
LIT=:MAR2
OUT-1=:MPCR
X

IFETCH:
X
X ROUTINE SETS UP PAR AND GOES TO A LOOP CHECKING GC1
X WHICH WILL BE SIGNAL TO DO I/O. RETURNS TO OTHER PROCESSOR
X AFTER CLEARING GC1
GETA00R-1=:CPCR
A3=:MIR
LIT=:MAR2
PAR=:LIT
OUTPUT1-1=:CPCR
RESET GC1
IOREQ-1=:MPCR
X

LOADERADDRESS: X SETUP ADDRESS TO LOAD PROGRAM INTO. ADDRESS WILL BE
X ADDED TO 1024 FOR CORRECT LOADING. NUMBER WILL BE WRITTEN
X TO BASE REGISTERS LOCATION COUNTER WILL BE STORED AT 20C
X CAN ALSO BE MODIFIED BY *CARD
X SET BASE REGISTER IN INCREMENTS OF 8192
X
GETA00R-1=:CPCR X LOCATION RETURNED IN A3
1 L=:MIR
COMP 10=:SAR;SBASE=:LIT
A3+=A3
LIT=:MAR2
1 L=:A1
7=:LIT;COMP 13=:SAR
LCTR=:SAVE
HW2=:BHI;IF SAI
WHEN SAI THEN A1+=MIR;INC
IF NOT COV THEN 8MAR+=1=:MAR2;JUMP
A3=:MIR;B
LIT=:MAR2
PL0C=:LIT
X ADDRESS OF LOCATION COUNTER
X A2 NOW HAS SECTOR ADDRESS/LOCATION COUNT
A2 R=:A2;CSAR
16=:SAR
A2 L=:A2
A2 OR B =:A2
HW2=:IF SAI
WHEN SAI THEN 0;STEP
X B REGISTERS 24-31 SET AND LOCATION COUNTER 106 SET
X
X
NEWWORD:
CARDREAD:
X ROUTINE USES A1,E,MIR
X THIS ROUTINE PLACES A 0/198 INTO MAILBOX

```



```

00C6 4809 0000 0000 0000 0000
00C7 0210 0000 0000 0000 0000
00C8 4809 2000 0000 0000 0000
00C9 0020 0000 0000 0000 0000
00EA 4809 0000 0000 0000 0000
00EB 0200 0000 0000 0000 0000
00EC 9408 0000 0000 0000 0000
00ED 0000 0000 0000 0000 0000
00EE 9809 2000 0000 0000 0000
00EF 0000 0000 0000 0000 0000
00F0 0200 0000 0000 0000 0000
00F1 0000 0000 0000 0000 0000
00F2 4809 0000 0000 0000 0000
00F3 8020 0000 0000 0000 0000
00F4 4809 2000 0000 0000 0000
00F5 4812 0000 0000 0000 0000
00F6 0030 0000 0000 0000 0000
00F7 9809 0000 0000 0000 0000
00F8 4809 0000 0000 0000 0000
00F9 9000 0000 0000 0000 0000
00FA 4809 0000 0000 0000 0000
00FB 4809 0000 0000 0000 0000
00FC 8A00 0000 0000 0000 0000
00FD 9C00 2000 0000 0000 0000
00FE 0040 0000 0000 0000 0000
00FF 0200 0000 0000 0000 0000
0100 4809 0000 0000 0000 0000
0101 4809 2000 0000 0000 0000
0102 0300 0000 0000 0000 0000
0103 6C00 2000 0000 0000 0000
0104 0020 0000 0000 0000 0000
0105 8C19 0000 0000 0000 0000
0106 0210 0000 0000 0000 0000
0107 4809 0000 0000 0000 0000
0108 0200 0000 0000 0000 0000
0109 1070 0000 0000 0000 0000
010A 0210 0000 0000 0000 0000

010B 1C88 0002 0000 0000 0000
010C 9809 0000 0000 0000 0000
010D 9F08 0000 0000 0000 0000
010E 0809 0000 0000 0000 0000
010F 0808 0000 0000 0000 0000
0110 8809 0000 0000 0000 0000
0111 AEC8 0000 0000 0000 0000
0112 4809 0000 0000 0000 0000
0113 682F 0000 0000 0000 0000

0114 4809 0000 0000 0000 0000
0115 4809 0000 1011 0000 0000

THIS SHOULD READ 20 WORDS INTO ADDRESS 334(8)
0=BC8,LCTR
33=LIT
LIT=HAR2
HEXADDR=LIT,COMP 16=HAR
B=HAR
ZBUFF: OUTPUT1-1=CPCR
IF NOT COV THEN B*HAR+1=HAR2,INC,STEP ELSE SKIP
ZBUFF-1=HPCR
LIT=:HAR,IF SAI
CAROFETCH:
CAROBUF=LIT
EXTOP-1=:CPCR
CAROFETCH-1=:HPCR
A2 L=A1
HEXADDR=LIT,COMP 20=HAR
LIT=HAR2
O=HAR,LCTR,SAVE
COMP 8=HAR,3=LIT
B L=HAR,INC,IF SAI
A1 R=BB1
COMP 3=HAR
A1 L=A1
B=HAR
IF COV THEN H*HAR+1=HAR2,STEP ELSE JUMP
WHEN SAI THEN LIT=HAR,INC,STEP
CAROBUF=LIT,24=HAR
INPUT-1=CPCR
B R=B
LIT EOL B
48=LIT,16=HAR
IF TRUE THEN LIT=HAR,STEP ELSE SKIP
HEXADDR=LIT,16=HAR
IF IRO THEN L=BB1,SKIP
CHECKCOL1-1=HPCR
B=HAR
CARDPRINT:
EXTOP-1=:CPCR
CARDPRINT-1=HPCR
CHECKCOL1-1=HPCR
EXTOP:
THIS ROUTINE PERFORMS REQUESTED EXTERNAL INTERFA
%TO THE IOP MIR = FUNCTION/ADDRESS
%RETURNS CONDITION OF OP IN B
SET GC2 WHEN GC2 THEN NOT 0=:HAR1
H*HAR,IF SAI
WHEN SAI THEN O,SET INT
IF INT
WHEN INT THEN STEP
H*HAR,IF ROC
WHEN ROC THEN BEX,RESET GC2
NOT B
IF ABT THEN JUMP ELSE RETN
%NO EXTERNAL FUNCTION
%
GETADDR:
%ASSUME HAR2 IS SET UP TO REREAD SAME WORD VIA BEX. RESULT WILL
%BE RETURNED IN A3, 7MIR CONTAINS ADDRESS OF WORD
BEX
O=:A3,BR2,LCTR
%FIRST DIGIT IS LAST BYTE OF B

```



```

0147 13C0 C000 0C00 C060 OUTPUT1-1=:CPCR
0148 4809 2001 2000 00F0 LIT L=:A2
0149 0060 00C0 0C00 C0E0 6=:LIT
014A 4809 C0C0 2000 00F0 A2 OR B=:A2
014B 4809 E0C1 1000 00F0 SETBUFF: A3 L=A3
014C E0C0 00C0 0C00 C0A0 COMP 16=:SAR,PRINTAREA-1=LIT
014D 4809 E15C 1C00 00F0 A3 OR LIT=:A3
014E 4809 0C00 0900 C0F0 0=:B0C8
014F 4809 0C40 0031 C0F0 B=MIR,LCTR
0150 021C 0000 0C00 C0E0 33=:LIT
0151 4809 2003 001C C0FC LIT=:MAR2
0152 0EEC 0000 0000 00EC PRINTAREA=LIT

ZEROL00P:
    OUTPUT1-1=:CPCR
    IF NOT COV THEN BRAR+1=:MAR2,INC; STEP ELSE SKIP
    ZEROL00P-1=:MPCR
    A3 R=:A1
    16=:SAR
    CONVERT: 0=:B0C8,LCTR
    1=:LIT;COMP 6=:SAR
    B L=:B
    B L=:MIR
    COMP 2=:SAR
    A1 R=:B01
    A1 L=:A1
    SET LC2;SAVE

CONVL00P:
    B L=:B
    COMP 5=:SAR
    B L=:MIR
    COMP 3=:SAR
    A1 R=:B01
    A1 L=:A1,INC
    IF COV THEN A3=:MAR2;SKIP ELSE STEP
    CONVL00P-1=:MPCR
    A3+1=:A3
    B=:MIR
    HW2;IF SA1
    WHEN SA1 THEN 0=:B0C8,LCTR
    3=:LIT
    IF LC2 THEN JUMP ELSE CALL
    %END OF LOOP ONE 32 BIT WORD WRITTEN IN 3 WORDS
    A2 L=:B
    COMP 16=:SAR,6=:LIT
    FETCHWORD-1=:AMPCR
    A2 R=:A2
    A2-1=:A2
    IF NOT M51 THEN A2 OR B C=:A2;JUMP
    LIT + B C=:A2
    PRINTBUFF-1=:MPCR
    FETCHWORD: A3+1=:A3
    A3+1 C=:A3,MAR2
    INPUT-1=:CPCR
    A2-1=:A2,CTR
    B=:A1,INC
    IF NOT COV THEN A3+1 C=:A3;STEP ELSE SKIP
    CONVERT-1=:MPCR
    LIT=:MAR2
    PLOC=LIT
    INPUT-1=:CPCR

0153 13C0 0000 0000 0060
0154 8408 0F46 001E 00F0
0155 152C 0000 0C00 C04C
0156 4809 E0C0 0C00 C0FC
0157 0000 0000 C0C0 0020
0158 4809 0000 C901 C0FC
0159 2010 0000 0C00 C08C
015A 4809 0C41 0B00 C0F0
015B 4809 0C41 0000 00F0
015C A00C 00C0 0C00 C030
015D 4809 A0C0 8F00 C0F0
015E 4809 A001 4000 00F0
015F 4852 0000 0000 C0F0

0160 4809 0C41 0B0C C0F0
0161 3000 00C0 0C00 C030
0162 4809 0C41 0C00 C0F0
0163 5000 00C0 0C00 C030
0164 4809 A0C0 8F00 00F0
0165 4809 A001 4002 00F0
0166 8C19 E0C3 001C C0F0
0167 15F0 0000 0C00 0040
0168 4809 E0C0 100C 00F0
0169 4809 0C40 0000 00F0
016A 5809 0000 0000 1C00
016B 9C08 00C0 C901 C0F0
016C 003C 00C0 0C00 00E0
016D 382E 00C0 0C00 00FC

016E 4809 C0C1 0B00 00F0
016F 0060 00C0 0C00 00AC
0170 0300 0000 000C 00C0
0171 4809 C0C0 A00C 00F0
0172 4809 C0DE 200C 00F0
0173 4429 C0D0 AC00 C0F0
0174 4809 2C41 AC00 C0FC
0175 031C 0000 0000 0040
0176 4809 E0C0 1C00 C0F0
0177 4809 E0C1 901C 00F0
0178 13E0 0000 0000 0060
0179 4809 C0DE 2005 C0F0
017A 4809 0C40 4002 C0FC
017B 5408 E0C1 9C00 C0F0
017C 157C 0000 000C C040
017D 4809 2003 001C C0FC
017E 0780 0000 0C00 00E0
017F 13EC 0000 000C 0060

04770C00 C
0478C000 D
0479C0C0 D
0480C000 C
0481C000 C
04820000 D
0483C0C0 D
0484C000 D
0485C0C0 D
0486C0C0 C
0487C0C0 D
0488C0C0 D
0489C0C0 D
0490C0C0 D
04910000 C
0492C0C0 C
0493C000 D
0494C0C0 D
0495C0C0 D
0496C0C0 D
0497C0C0 D
0498C0C0 C
0499C0C0 D
0500C0C0 D
0501C0C0 D
0502C0C0 D
0503C0C0 D
0504C0C0 D
0505C0C0 D
0506C0C0 D
0507C0C0 D
0508C0C0 D
0509C0C0 D
0510C0C0 D
0511C0C0 D
0512C0C0 C
0513C0C0 D
0514C0C0 C
0515C0C0 D
0516C0C0 C
0517C0C0 D
0518C0C0 D
0519C0C0 D
0520C0C0 D
0521C0C0 D
0522C0C0 D
0523C0C0 D
0524C0C0 D
0525C0C0 D
0526C0C0 D
0527C0C0 D
0528C0C0 D
0529C0C0 D
0530C0C0 D
0531C0C0 D
0532C0C0 D
0533C0C0 D
0534C0C0 D
0535C0C0 D
0536C0C0 D

```



```

0180 4809 0C40 2030 C0F0
0181 0E50 00C0 0030 C0C0
0182 2029 0000 0C30 00F0
0183 0320 0000 0030 C040

0184 4809 00C1 0B30 C0F0
0185 0EE0 00C0 0030 00AC
0186 4809 2C50 0C30 C0F0
0187 10A0 00C0 0030 006C
0188 1830 00C0 0030 C040
0189 4809 EPC0 9C00 C0F0
018A 14A0 0000 0030 C040

018B 49C9 20C3 0C1C C0F0
018C 0060 0000 0030 00E0
018D 4809 00C0 0C31 00F0
018E 0030 0000 0030 C0EC
018F 13E0 0000 0030 006C
0190 4812 0C40 4D30 C0F0
0191 4809 A0C1 4030 C0F0
0192 8000 0000 0030 0030
0193 4809 A0C0 9030 C0F0
0194 4809 A0C1 4C30 C0F0
0195 4809 EC41 1830 C0F0
0196 900C 00C0 0000 C030
0197 4809 0C41 0B30 00F0
0198 8000 00C3 0C00 C030
0199 4809 EC40 0B30 C0F0
019A 8C00 0F46 001C 00F0
019B 002C 0000 0030 00EC
019C 2C0E 0000 0021 00F0
019D 18E0 00C0 0030 0040
019E 4809 ACC1 4030 00F0
019F 8000 0000 0030 0030
01A0 4809 A000 9C30 C0F0
01A1 4809 EC40 4B30 C0F0
01A2 380B 00C0 0C30 C0F0
01A3 0330 0000 0030 0040
01A4 4809 20C3 0C1C C0F0
01A5 0D4C 0003 0030 C0E0
01A6 13EC 00C0 0030 C060
01A7 113C 00C0 0000 0060
01A8 4809 A000 0030 C0FC
01A9 4809 E0C2 0030 C0F0
01AA 680B 0000 0030 C0F0
01AB 0B2C 00C0 0030 C040
01AC 4809 00C1 CB30 00FC
01AD A00C 0000 00C0 002C
01AE 4809 EC40 061C C0FC
01AF 13C0 0000 0030 0060
01B0 0E50 0000 0030 C040
01B1 0D88 0018 001C C0F0
01B2 13E0 C0C0 0030 0060
01B3 4809 0C40 C030 C0F0
01B4 0000 0000 0030 002C
01B5 4809 A640 0030 00F0
01B6 0340 00C0 0000 00CC
01B7 4809 0000 0030 00F0
01B8 4824 0C40 101C C0FC

B=:A2
NEWRO-1=:AMPCR
IF NOT LC1 THEN JUMP
I0RETURN-1=:MPCR
%
PRINTBUFF:
1 L=:B
COMP 16=:SAR;PRINTAREA=LIT
LIT OR B=:MIR
EXTOP-1=:CPCR
PRINTBUFF-1=:MPCR
A3 R=A3
SETBUFF-1=:MPCR

DATA:
LIT=:HAR2; SET LC1
COL9X12=LIT
O=:MIR;LC1R
3=LIT
INPUT-1=:CPCR
B=:A1;B1;SAVE
A1 L=:A1
COMP 4=:SAR
A1 R=:A3
A1 L=:A1
A3 + B L=:A3;BAD
COMP 3=:SAR
B L=:B
COMP 1=:SAR
A3+D=:B;MIR;INC
IF COV THEN BHAR+1=:HAR2; STEP ELSE JUMP
2=LIT
IF LC1 THEN LC1R; STEP ELSE SKIP
DATA1-1=:MPCR
A1 L=:A1
COMP 4=:SAR
A1 R=:A3
A3+B=:A1;B
IF LC2 THEN STEP ELSE SKIP
RESERVE1-1=:MPCR
LIT=:HAR2
CARDBUF=LIT
INPUT-1=:CPCR
GETADDR-1=:CPCR
A1=:MIR
NOT A3
IF AB1 THEN STEP ELSE SKIP
NOPACK-1=:MPCR
1 L=:B
COMP 10=:SAR
A3+B=:HAR2
OUTPUT1-1=:CPCR
NEWRO-1=:MPCR
I0REQ:
SET GC1; WHEN GC1 THEN B11=:HAR2
INPUT-1=:CPCR
B R=:A1
COMP 16=:SAR
A1+AMPCR=:AMPCR
EXTI0REQ-1=:AMPCR
STEP
B=:HAR2;A3;EXEC
%RESTORE A2 FROM SAVED LOCATION
%6=:1/ADDRESS FOR PRINT BUFFER CODE
%INPUT DECIMAL DATA
%GET RID OF BCL HEADER
%ISOLATE LOW ORDER 4 BITS
%MULT BY 8
%MULT BY 2
%COULD HAVE BEEN CALLED BY RESERVE
%REFREAD COLS 4-8 FOR ADDRESS
%IF ADDR=:0 PUT IN LINE
%OTHERWISE PUT AT ADDRESS GIVEN
%OFFESET ADDRESS BY 1024
%NORMAL RETURN TO GET NEXT CARD
%

```



```

01B9 49F6 0000 0000 00F0
01BA 4949 0019 0005 00F0
01BB 8808 0000 0002 00F0
01BC 1800 0000 0000 004C
01BD 0350 0000 0000 00CC
01BE 0360 0000 0000 00C0
01BF 0370 0000 0000 00C0
01C0 0380 0000 0000 00C0
01C1 0390 0000 0000 00C0
01C2 03A0 0000 0000 00C0
01C3 03B0 0000 0000 00C0
01C4 03CC 0000 0000 00C0
01C5 000C 0000 0000 00CC
01C6 4809 0000 0000 00F0
01C7 1F4C 0000 0000 00C0
01C8 4809 0000 0000 00F0
01C9 4809 0640 0000 00F0
01CA 4809 2001 0000 00F0
01CB 001C 0000 0000 00A0
01CC 4809 0C40 0000 00F0
01CD 10AC 0000 0000 0060
01CE 1CCC 0000 0000 0040
01CF 141C 0000 0000 0040
01D0 1410 0000 0000 004C
01D1 4809 E3C1 1000 00F0
01D2 000C 0000 0000 0020
01D3 2809 E0C0 8000 00F0
01D4 10A0 0000 0000 006C
01D5 1D3C 0000 0000 0040
01D6 1B90 0000 0000 0040
01D7 4800 0000 0000 00F0
01D8 4809 0000 0000 00F0
01D9 021C 0000 0000 00E0
01DA 4809 20C0 001C 00F0
01DB 0EE0 0000 0000 00E0
01DC 4812 0C40 0000 00F0
01DD 9809 0000 0000 1CF0
01DE 9C08 0000 0000 00F0
01DF 8429 0F46 001E 00F0
01E0 4809 E157 1800 00F0
01E1 9070 0000 0000 0080
01E2 4809 0C41 0B01 00F0
01E3 809C 0000 0000 0080
01E4 4809 EC40 100C 00F0
01E5 4809 E640 401C 00F0
01E6 118C 0000 0000 00C0
01E7 4809 20C0 1C00 00F0
01E8 0E00 0000 0000 00E0
01E9 13E0 0000 0000 0060
01EA 4809 0C40 0000 00F0
01EB 4809 E0C0 1C1E 00F0

IORETURN:
  RESET GC1, B111=CTR
  INC, WHEN COV THEN STEP
  IOREQ-1=MPCR
EXTIOREQ:
  DUMPREG-1=AMPCR
  DUMPAD-1=AMPCR
  CARDIO-1=AMPCR
  DISKIO-1=AMPCR
  DISKWRITE-1=AMPCR
  PRINTERIO-1=AMPCR
  WORDIO-1=AMPCR
  PRINTLINE-1=AMPCR
  START-1=AMPCR
DUMPREG:
  STEP
  BLANKLINE=AMPCR
  STEP
  AMPCR=MIR
  LIT L=BB1
  16=SAR,1=LIT
  B=MIR
  OUTBLANK: EXTOP-1=CPCR
  OUTBLANK-1=MPCR
  IODUMP-1=MPCR
DUMPAD:
  IODUMP-1=MPCR
CARDIO:
  READ ONE CARD INTO BUFFER ADDRESSED IN B,MAR2
  A3 L=A3
  COMP 16=SAR
  A3 R=MIR,IF LC1
CARDIO1:
  EXTOP-1=CPCR
  CARDIO1-1=MPCR
  IORETURN-1=MPCR
DISKWRITE:
  WAIT
PRINTERIO:
  O=BC8,LCTR
  33=LIT
  LIT=MAR2. PRINTAREA=LIT
  B=MIR,SAVE
  MW2,IF SAI
  WHEN SAI THEN 0,STEP
  IF NOT COV THEN BHAR+1=MAR2,INC,JUMP
  A3 AND LIT L=A3,BAD
  COMP 3=SAR,7=LIT
  B L=B,LCTR
  COMP 1=SAR,9=LIT
  A3+B=A3
  A3+AMPCR=MAR2,A1
  ERRORLIST=AMPCR
  LIT=A3
  PRINTAREA-1=LIT
  ERROR1: INPUT-1=CPCR
  B=MIR
  A3+1=A3,MAR2,INC
0597C000 D
0598C000 D
0599C000 C
0600C00C C
0601C000 D
0602C000 D
0603C000 D
0604C000 F
0605C000 D
0606C000 D
0607C000 D
0608C000 D
0609C000 D
0610C00C D
0611C000 C
0612C000 D
0613C000 D
0614C00C D
0615C000 D
0616C00C D
0617C000 D
0618C000 D
0619C000 D
0620C00C D
0621C00C D
0622C000 D
0623C00C D
0624C00C D
0625C000 D
0626C000 D
0627C000 D
0628C000 C
0629C000 D
0630C000 D
0631C000 D
0632C000 C
0633C000 D
0634C000 C
0635C000 D
0636C000 C
0637C000 D
0638C000 D
0639C00C D
0640C000 C
0641C000 D
0642C000 D
0643C000 D
0644C000 D
0645C000 D
0646C000 D
0647C000 C
0648C000 C
0649C000 D
0650C00C D
0651C000 D
0652C000 D
0653C00C D
0654C000 D
0655C000 D

```



```

01EC 13CC 0000 0000 C060
01ED 8408 A0C3 401C C0F0
01EE 1E80 0C00 0030 004C
01EF 4809 00C1 0B30 C0F0
01F0 0E00 0000 0030 C0A0
01F1 4809 2C5C 0030 C0FC
01F2 10A0 00C0 0030 C06C
01F3 1F1C 0000 0070 C040
01F4 1B9C 0000 0030 00C0
01F5 2829 0000 0030 00F0
01F6 0E50 0000 0030 004C

01F7 13E0 0000 0030 0060
01F8 4809 0000 001C C0F0
01F9 4809 0C40 0030 C0F0
01FA 13C0 0000 0000 0060
01FB 4809 E001 0B30 00F0
01FC 00E0 0000 0030 00AC
01FD 4809 2C5D 8080 C0F0

01FE 10AC 0000 0030 006C
01FF 1F0C 0000 0030 004C
0200 1B9C 00C0 000C C04C

0201 2809 20C1 1000 C0FC
0202 006C 00C0 0030 00A0
0203 4809 EF4C 0030 C0F0
0204 10A0 0000 0000 0060
0205 2030 00C0 0000 C040
0206 1B9C 0000 0030 C040

0207 2809 0C40 0030 00F0
0208 10AC 00C0 0000 0060
0209 2070 0000 0030 004C
020A 1B90 0000 0030 C040
020B 4849 00C0 0030 C0F0
020C 18A0 0000 0030 C040
020D 4809 CC40 2030 C0F0
020E 4809 2003 001C CCFO
020F 0780 00C0 0000 C0E0
0210 0E30 0000 0000 0040

0211 1130 0000 0030 0060
0212 4809 2003 401C 00F0
0213 AD6C 00C0 0030 00A0
0214 4809 0013 8030 00F0
0215 3809 E002 0030 00F0
0216 6E59 C00E 1000 C0F0
0217 4809 EC40 1C30 00FC
0218 13EC 0000 0030 0060
0219 0030 0000 0030 C0E0
021A 4812 00C0 0001 C0F0
021B 4809 0C41 8B90 00F0
021C 03F0 0000 0030 C0B0
021D 4809 2C56 001C C0F0
021E 4809 2F52 0030 00F0
021F 6419 0000 0030 C0F0
0220 0300 0000 0030 C040

OUTPUT1-1=CPCR
IF NOT COV THEN A1+1=A1,MAR2;STEP ELSE SKIP
ERROR1-1=MPCR
1 L=B
COMP 16=SAR;PRINTAREA=LIT
LIT OR B=MIR
ERROR2: EXTOP-1=CPCR
ERROR2-1=MPCR
IORETURN-1=AMPCR
IF LC1 THEN JUMP ELSE STEP
NEWROD-1=MPCR

WORD10:
INPUT-1=CPCR
B110=MAR2
B=MIR
OUTPUT1-1=CPCR
A3 L=B
COMP 16=SAR;14=LIT
LIT OR B C=MIR

WORD101:
EXTOP-1=CPCR
WORD101-1=MPCR
IORETURN-1=MPCR
PRINTLINE: %PRINT ONE LINE FROM BUFFER ADDRESSED IN B
LIT L=A3;IF LC1
COMP 16=SAR; 6=LIT
A3 XOR BMAR=MIR
PRINTLINE1: EXTOP-1=CPCR
PRINTLINE1-1=MPCR
IORETURN-1=MPCR
DISK10: %READ DISK AS IN MAILBOX B CONTAINS CODE/ADDRESS
B=MIR;IF LC1
DISKERR: EXTOP-1=CPCR
DISKERR-1=MPCR
IORETURN-1=MPCR
RESERVE: SET LC2
RESERVE1: DATA-1=MPCR
RESERVE1: A2+B=A2
LIT=MAR2
PL0C=LIT
OUT-1=MPCR
WORD: %W - ASSUME CHARACTER INPUT UNTIL FIRST QUOTE
%ELSE USE ENTIRE CARD AS CHARACTER STRING. USE ADDRESS
%IN COLS 4-8 ELSE NEXT LOCATION IF ZERO.
GETADDR-1=CPCR
LIT =A1,MAR2
COMP 10=SAR;COL9X12=LIT
B111 R=B
NOT A3; IF LC2
IF ABT THEN A2-1=A3; SET LC2; SKIP
A3 +B=A3
WORD1: INPUT-1=CPCR
3=LIT
LCIR;SAVE
B C=B,MIR
63=LIT;COMP 8=SAR
LIT AND B=MAR2
LIT EOL BMAR
IF FALSE THEN INC; SKIP ELSE STEP
WORD2-1=MPCR
%FECH WORD TO BE PRINTED
%WRITE WORD OUT TO MAILBOX+1
%GET ADDRESS INTO UPPER B
%SEND 14/ADDRESS
%RESERVE AMOUNT OF SPACE IN COLS.9-1602910000 D
%INCREMENT PROGRAM COUNTER
%SET UP FOR 4 CHARACTERS
%GET NEXT COLUMN
%MASK LEAST 6 BITS
%SAVE LEAST 6 BITS
%SEE IF ITS A QUOTF "

```


07160000 0
07170000 0
07180000 0
07190000 0
07200000 0
07210000 0
07220000 0
07230000 0
07240000 0
07250000 0
07260000 0

IF COV THEN A3+1=HAR2,A3, STEP ELSE JUMP
IF LC2 THEN A3=A2,SET LC2 #USE NEXT ADDRESS FROM A2
OUTPUT1-1=CPCR
A1+1=A1,HAR2
WORD1-1=AMPCR
A1 EOL LIT
LASTCOL=LIT
IF FALSE THEN JUMP
WORD2: IF LC2 THEN A3+1=A2
NEWWORD-1=HPCR
END
#218 = LAST COLUMN OF CARD
#RESET PROGRAM COUNTER

0221 8C0D EDC3 1C1C C0F0
C222 3E49 E0C0 203C C0F0
C223 13C0 0000 0090 0060
0224 4809 A0C3 401C C0F0
0225 2170 0000 0C20 00CC
0226 4809 A152 0C2C 00F0
C227 0E80 0000 007C CCE0
C228 6029 0000 0030 00F0
0229 3C09 ECC3 203C C0F0
C22A 0E50 0000 0000 C040
C22B 00C0 00C0 0C3C C040
0220 228C 00C0 0070 C040
C1C4 2000 0000 0C30 00C0
C1C3 1F60 0000 0C3C C0C0
01C2 1D70 0000 007C C0CC
01C1 106C 0000 0C3C 00CC
01C0 206C 0000 0C3C C0CC
01BF 100C 0000 007C 00C0
01BE 1CF0 00C0 003C C0CC
01B0 1C50 00C0 0C00 00CC
01B6 18CC 0000 0030 00C0
01A3 20C0 00C0 0030 0040
0183 1890 00C0 0000 C040
0175 183C 0000 0C00 0040
0170 1750 0000 0000 C0C0
0126 13E0 0000 0030 C060
0108 10AC 00C0 0C00 C060
0CFF 13EC 00C0 0030 0060
0CF0 10A0 0000 007C 0060
CCEB 13C0 00C0 0C3C 006C
0CD2 1130 00C0 0C30 C060
0CD1 18D0 0000 0030 C04C
0CCF 13C0 0000 003C C060
0CCB 1130 00C0 0030 0060
CCEA 0E30 0000 003C C04C
0CC6 1130 00C0 0C30 C060
0CC5 123C 00C0 0C00 0040
0CC4 113C 0000 0030 C06C
0CC3 1230 00C0 0000 0040
0CC0 1130 0000 0000 006C
CCBF 0E30 0000 0C30 C040
0CB9 1130 0000 003C C060
0CB5 0E30 0000 0000 004C
0CB0 0E30 0000 0C3C 004C
CCAC 080C 00C0 003C 00C0
0CA6 0B2C 00C0 0000 0040
0CA3 0B50 00C0 0C30 006C
0C90 0B5C 00C0 0000 C060
0083 0B5C 00C0 0C30 C060
0CB1 0A6C 00C0 0000 C04C
0C75 0B5C 0000 0030 C06C
0C73 0A60 0000 007C 0040
CC62 0B50 00C0 0030 0060
0C60 0730 0000 003C 0040
CC50 073C 0000 0C30 C040
0C5A 0B10 00C0 0C30 0040
0C40 0B50 00C0 0000 0060
0C4A 107C 0000 0C30 C040
0C49 210C 00C0 0000 0040
CC46 20A0 00C0 0000 0040


```

0043 1400 0000 0000 0000 0000
0040 0010 0000 0000 0000 0000
0030 00A0 0000 0000 0000 0000
003A 0050 0000 0000 0000 0000
0037 0030 0000 0000 0000 0000
0034 0080 0000 0000 0000 0000
0031 00F0 0000 0000 0000 0000
002E 0040 0000 0000 0000 0000
002B 10A0 0000 0000 0000 0000
0024 13E0 0000 0000 0000 0000
0021 0020 0000 0000 0000 0000
0015 10A0 0000 0000 0000 0000
000F 1300 0000 0000 0000 0000
C ERRORS, 728 CARDS. 3969 TOKENS. 6700 RULES. 507 SECONDS.
# OUP FILE ON DISK:UYK7-LOADER # 1:1209.1 #####
INVALID ARGUMENT: FILE10 4 #####

```


APPENDIX D. SAMPLE AN/UYK-7 SORT PROGRAM

This appendix provides a copy of a sample program which was used to demonstrate the capability of the D-Machine to function as an AN/UYK-7. The program, written in AN/UYK-7 Machine Language, will read twelve numbers per card, print the numbers, sort the input and print the sorted results. Any number of cards could be read and sorted until the "END" card is encountered. At this point the program would reinitialize memory and be ready to accept the next program. Several of the Loader's macro functions, including the L, W, D and N options, were utilized along with the appropriate formats for the five instruction types. It must be emphasized that when this program was run the ALGOL Machine had been removed and the D-Machine reconfigured through microprogramming to be an AN/UYK-7. The entire output format and program execution were produced by the Loader/Emulator combination with the IOP being used strictly as an Input/Output Channel. Examples of the Loader and Debugger output optional listings for this sample program are presented in Appendix E and F.


```

L 00002      ORIGIN SORT ROUTINE AT LOCATION 0002
D 01001      10
203320001001
110310001002
440310001002
532220000014
523220000003
102320001000
512420000021
320020001000
201320000777
514020000002
101310001002
241310001001
240310001002
330020001000
514020000006
201320001000
530000004040      JUMP TO TITLE ENDING ROUTINE
O 04000      BEGIN TITLE PRINTING ROUTINE AT 04000
071400005000
071400005352
071400005352
070400001022
071400005044
071400005352
071400005110
071400005352
071400005154
070410001002
106320001002
446320005416
531220004060      JUMP TO COMPLETION ROUTINE IF END CARD
071400005352
071400005220
071400005352
071400005352
071410001002
510000000002      JUMP TO BEGINNING OF SORT ROUTINE
O 04040      RETURN HERE FROM SORT ROUTINE AND FINISH TITLES
071400005352
071400005352
071400005264
071400005352
071400005330
071400005352
071400005352
071420001002
071400005352
071400005352
510000004011
O 04060      START COMPLETION ROUTINES HERE
071400005352
071400001002      PRINT END
7706000      TERMINATE PROGRAM AND RESTART EMULATOR

```



```

W 05000          AN/UYK-7  EMULATION  DEMONSTRATION  PROGRAM
W 05022
W 05044          THIS IS A TEST OF THE AN/UYK-7  EMULATION
W 05066
W 05110          USING A MACHINE LANGUAGE PROGRAM
W 05132
W 05154          FOR A DEMONSTRATION SORT ROUTINE USING
W 05176
W 05220          THE FOLLOWING INPUT NUMBERS:
W 05242
W 05264          THE RESULTS OF THIS SORT
W 05306
W 05330          ARE AS FOLLOWS:
W 05352
W 05374
W 05416END
N 04000

```

```

123 456 789 987 654 321 023 456 875 888 555 213
357 652 389 123 583 742 928 654 987 248 965 281
321 654 987 123 456 789 369 258 147 963 852 741
END      TERMINATE SORT ROUTINE, READY FOR NEXT PROGRAM

```


APPENDIX E. SAMPLE LOADER OUTPUT LISTING

This appendix provides a copy of the output received from the Loader when the sample program discussed in Appendix D is run. This output is obtained by turning on the IRQ switch on the Loader's Interpreter. The output serves as a hard copy program listing for the programmer and can be utilized for debugging. The address to the left of each instruction is the octal assignment for that instruction, derived from the last "O" or "L" card, and was offset by 1024. The Loader input is terminated by an "N" card which initializes the PAR and signals the Emulator to commence execution.

| | |
|-----------|---------------------------------------|
| L ' 00002 | ORIGIN SORT ROUTINE AT LOCATION 0002 |
| D 01001 | 10 |
| 2002 | 203320001001 |
| 2003 | 110310001002 |
| 2004 | 440310001002 |
| 2005 | 532220000014 |
| 2006 | 523220000003 |
| 2007 | 102320001000 |
| 2010 | 512420000021 |
| 2011 | 320020001000 |
| 2012 | 201320000777 |
| 2013 | 514020000002 |
| 2014 | 101310001002 |
| 2015 | 241310001001 |
| 2016 | 240310001002 |
| 2017 | 330020001000 |
| 2020 | 514020000006 |
| 2021 | 201320001000 |
| 2022 | 530000000040 |
| 0 04000 | BEGIN TITLE PRINTING ROUTINE AT 04000 |
| 6000 | 0714000003000 |
| 6001 | 0714000003352 |
| 6002 | 0714000003352 |
| 6003 | 070420001022 |
| 6004 | 0714000005044 |
| 6005 | 0714000003352 |
| 6006 | 0714000003110 |
| 6007 | 0714000003352 |
| 6010 | 0714000003154 |
| 6011 | 070410001002 |
| 6012 | 106320001002 |
| 6013 | 4463200003416 |
| 6014 | 5312200000060 |
| 6015 | 0714000003352 |
| 6016 | 0714000003220 |
| 6017 | 0714000003352 |
| 6020 | 0714000003352 |
| 6021 | 071410001002 |
| 6022 | 5100000000002 |
| 0 04040 | RETURN HERE |
| 6040 | 0714000003352 |
| 6041 | 0714000003352 |
| 6042 | 0714000005264 |
| 6043 | 0714000003352 |
| 6044 | 0714000003330 |
| 6045 | 0714000003352 |
| 6046 | 0714000003352 |
| 6047 | 071420001002 |
| 6050 | 0714000003352 |
| 6051 | 0714000003352 |
| 6052 | 5100000000011 |

JUMP TO TITLE ENDING ROUTINE

JUMP TO TITLE PRINTING ROUTINE AT 04000

JUMP TO COMPLETION ROUTINE IF END CARD

JUMP TO BEGINNING OF SORT ROUTINE
FROM SORT ROUTINE AND FINISH TITLES


```

0 04060
6C60 071400003352
6C61 07143000C1002 PRINT END
6062 7706000 AN/UYK-7 EMULATION DEMONSTRATION PROGRAM
W 05000
W 05022
W 05044
W 05066
W 05110
W 05132
W 05154
W 05176
W 05220
W 05242
W 05264
W 05306
W 05330
W 05352
W 05374
W 05416END

START COMPLETION ROUTINES HERE

071400003352
07143000C1002 PRINT END
7706000 AN/UYK-7 EMULATION DEMONSTRATION PROGRAM

THIS IS A TEST OF THE AN/UYK-7 EMULATION
USING A MACHINE LANGUAGE PROGRAM
FOR A DEMONSTRATION SORT ROUTINE USING
THE FOLLOWING INPUT NUMBERS?

THE RESULTS OF THIS SORT
ARE AS FOLLOWS?

```


N 04000 AN/UYK-7 EMULATION DEMONSTRATION PROGRAM

THIS IS A TEST OF THE AN/UYK-7 EMULATION
USING A MACHINE LANGUAGE PROGRAM
FOR A DEMONSTRATION SORT ROUTINE USING
THE FOLLOWING INPUT NUMBERS:

123 456 789 987 654 321 023 456 875 888 555 213

THE RESULTS OF THIS SORT
ARE AS FOLLOWS:

023 123 213 321 456 456 555 654 789 875 888 987

THE FOLLOWING INPUT NUMBERS:

357 652 389 123 583 742 928 654 987 248 965 281

THE RESULTS OF THIS SORT
ARE AS FOLLOWS:

123 248 281 337 359 583 652 654 742 928 965 987

THE FOLLOWING INPUT NUMBERS:

321 654 987 123 456 789 359 258 147 963 852 741

THE RESULTS OF THIS SORT
ARE AS FOLLOWS:

123 147 258 321 359 456 654 741 789 852 963 987

END TERMINATE SORT ROUTINE, READY FOR NEXT PROGRAM

APPENDIX F. SAMPLE DEBUGGER OUTPUT LISTING

This appendix provides a sample of the output received from the Loader when it was used as a debugger for the sample program discussed in Appendix D. This output is obtained by turning on the IRQ switch on the Emulator's Interpreter. The switch was not left set for the entire program's execution because the output requires approximately thirty pages. The IRQ switch causes all CMR registers from address 0000 to 0035 to be printed as each instruction is executed. Each time an instruction does a Y-Operand write that address is also dumped. The addresses are printed in the leftmost column and include the following information which can be used during program debugging:

| OCTAL ADDRESS | DESCRIPTION |
|------------------|---|
| 0-7 | Task A-Register contents |
| 10 | Unused (always = 0) |
| 11-17 | Task Index B-Register contents |
| 20-27 | Task Base S-Register contents |
| 30 | Repeat Instruction (if applicable) |
| 31 | Current Instruction being Repeated (if applicable) |
| 32 | Current Indirect Control Word (if applicable) |
| 33 | Y-Operand for Indirection (if applicable) |
| 34 | ICW address in memory (if applicable) |
| 35 | Program Address Register |

Those addresses marked "if applicable" would normally be zero, but could contain data if either the Repeat mode or

Indirection mode had been used previously in the program. The lower 20 bits of address 0035 (PAR) point to the octal address of the next instruction. The upper 12 bits of the PAR are those ASR bits which were implemented as discussed in Section C of Chapter V. It is noteworthy that the output is an eleven bit octal representation of a 32-bit binary word, thus the leftmost octal digit represents only two bits.

By utilizing this optional listing in combination with the Loader listing, the programmer should be capable of proceeding through his program step by step. This facility proved an invaluable aid in troubleshooting the Emulator as each new instruction was written and tested.

165

GLOSSARY

Active Status Register (ASR): A special word in the AN/UYK-7 used to indicate the status of special conditions or modes of operation. Each bit of the word has a separate meaning to the central processor.

ADO: The Burrough's Advanced Design Organization, Paoli, Pennsylvania, which designed the Naval Postgraduate School's D-Machine.

A Registers (A1, A2, A3): Each of the three A registers is functionally identical. The A registers are used for temporary data storage within the Logic Unit of the Interpreter and serve as a primary input to the adder.

Adder: The adder in the Logic Unit of the Interpreter, is a modified version of a straightforward carry lookahead adder. It is also used for executing logic operations.

Alternate Microprogram Count Register (AMPCR): The AMPCR is a 12-bit register in the Memory Control Unit of the Interpreter, which contains the jump or return address for program jumps and subroutine returns within a microprogram.

AMPCR: Alternate Microprogram Count Register.

Arithmetic (A) Registers: There are two sets of eight Arithmetic Registers in the AN/UYK-7. They are used extensively in arithmetic calculations and are directly addressable by the programmer.

ASR: The Active Status Word in the AN/UYK-7.

B Register: The B register is the primary interface between the Logic Unit of the Interpreter and the Data/Program Memory or Devices through the Switch Interlock. It also serves as a secondary input to the adder.

Barrel Switch: The barrel switch is a matrix of gates in the Logic Unit of the Interpreter, used to shift a parallel data word any number of places to the left or right in a single clock time.

Base Registers 1 and 2 (BR1, BR2): The Base Registers are two 8-bit registers in the Memory Control Unit of the Interpreter, which usually contain the base address of a 256-word block of Data/Program Memory.

Base (S) Registers: There are two sets of eight Base Registers in the AN/UYK-7 used extensively in multi-programming and multi-processing. Base registers are used for operand address calculations.

BR1 and BR2: Base Registers 1 and 2 in the Interpreter.

Building Block: The term used to describe the primary functional units of the Interpreter Based System and includes: Interpreter, Data/Program Memory and Switch Interlock.

Control Memory Registers (CMR): A block of 89 registers in the AN/UYK-7 used for special fast memory operations.

Condition Register (COND): The COND is a 12-bit register in the Control Unit of the Interpreter and is used to store various condition bits for use during program execution.

Central Processing Unit (CPU): The primary arithmetic and control unit in a conventional computer system.

Condition Select: The condition select is a matrix of gates in the Control Unit of the Interpreter that computes the results of a computation or logic operation in the Logic Unit with a preselected result. The results of the comparison may be used to determine the sequence of execution of microprogram instructions.

Control Unit (CU): The CU, one of the five functional units of the Interpreter, is used for condition testing and the storage and distribution of enable signals received from the nanoinstructions.

Counter (CTR): The CTR is an 8-bit counter in the Z register section of the Memory Control Unit of the Interpreter, used for loop control and other counting functions.

CTR: Counter in the Interpreter.

Data/Program Memory: The Data/Program Memory of the Interpreter, also called S-Memory, provides storage for data and program, and functions similarly to the main memory modules of a conventional computer system.

Emulation: Describes a process through which the hardware components of one machine (Host) are made to "appear" to assume the specific characteristics of the hardware of another machine (target).

End-Around-Shift: A shift operation in either the left or right direction, in which the bit or bits which would be shifted out of the register are reinserted in the more significant end.

End-Off-Shift: A shift operation in either the left or right direction, in which the bit or bits shifted out of the register are lost. Vacated bit positions are automatically replaced by zeros.

Firmware: In the Interpreter Based System, firmware is the combination of stored logic in the micro-memory and the hardware logic of the Interpreter.

GC Bits: A set of two (GC1, GC2) global condition bits in the Interpreter. They are used as lockouts during communications to the I/O processor.

HALFFETCH: A subroutine written for the Emulator, which is executed when a halfword instruction has been decoded.

Horizontal: A type of microprogramming instruction which controls multiple gates simultaneously allowing parallel functions to execute.

Host: A term used in Emulation to define the machine on which another machine is to be emulated (imitated). In this thesis the Burrough's D-Machine.

ICW: Indirect Control Word in the AN/UYK-7.

IFEICH: A subroutine written for the Emulator, which reads the next instruction from main memory, deciphers the Op-code and passes control to the appropriate Op-code execution subroutine.

Incrementer (INCR): The INCR is in the Memory Control Unit of the Interpreter and increments the address of the next microinstruction to be executed by the Interpreter by zero, one or two, depending on the successor instruction which is either implied or specified. A WAIT causes an increment by zero, a STEP causes an increment by one, and a RETN causes an increment by two.

Indirection: An addressing technique or mode of operation of the AN/UYK-7 in which Y-Operand address calculation points to an Indirect Control Word (ICW) which in turn points to the operand or another ICW. Indirection is determined by the i-field of an instruction.

Indirect Control Word (ICW): The ICW is a 32-bit word in the AN/UYK-7 used in Indirection which can be broken down into several fields. The fields determine the mode of indirect address calculation to be used in pointing to the next ICW or the Y-Operand.

Indexing: An addressing technique wherein the normal address calculated is incremented/decremented (indexed) by the value in the specified index register.

Index (B) Register: There are two sets of seven Index Registers in the AN/UYK-7. They provide the ability to perform Indexing during memory referencing, and may also be used as counters.

INT: An Interrupt signal issued by an Interpreter.

Interpreter: The Interpreter is the basic building block of the Interpreter-Based System. Functionally, it is characterized by the combination of microprogram instructions stored in its M memory and the combination of bits in its nanoinstruction which enable signals to implement the hardware logic.

Interpreter-Based System: A computer design concept that provides, in the form of basic building blocks, the throughput and flexibility for a variety of data processing requirements.

Interrupt: In the AN/UYK-7, the Interrupt signals the central processor to cease its normal instruction flow and respond to a request for services. In the D-Machine, Interrupt (INT) is a flag which may be set between processors signalling that a message has been placed in the Mailbox.

Least Significant Bit (LSB): For a number or value represented in binary notation, that bit position which represents the least significant portion of the number.

LIT: Literal Register in the Interpreter.

Literal Register (LIT): An 8-bit register in the Z section of the Memory Control Unit of the Interpreter, which is used for temporary storage of literals from microinstructions.

Logic Unit (LU): The LU is one of the five major functional units of the Interpreter. It performs all of the arithmetic, Boolean logic, and shifting operations of the Interpreter.

Mailbox: An address (64K) in the Interpreter's S-Memory which is used for passing messages between Interpreters and the IOP.

MAR: Memory Address Register in the Interpreter.

Memory Address Register (MAR): The MAR is an 8-bit register in the Memory Control Unit of the Interpreter, which contains the least significant 8 bits of a memory or device address.

Memory Control Unit (MCU): The MCU is one of the five major functional units of the Interpreter. It controls the sequence of execution for microinstructions; the addressing of Data/Program Memory; and the selection of devices.

Memory Information Register (MIR): The MIR is a register in the Logic Unit of the Interpreter which serves as the output interface register between the Interpreter and the Switch Interlock.

Microprogram Address Control Register (MPAD CNTL): The MPAD CNTL, a register in the Memory Control Unit of the Interpreter - controls the loading of the MPCR and the AMPCR, and determines the value of the increment.

Microprogram Count Register (MPCR): The MPCR, located in the Memory Control Unit of the Interpreter, is a 12-bit register that usually contains the address, in M-memory, of the microinstruction presently being executed by the Interpreter.

Microprogram Memory (M-Memory): The M-Memory is one of the five major functional units of the Interpreter. It stores microinstructions which characterize the Interpreter for a given application, and may be implemented as a read/write semiconductor memory.

Microprogramming: A technique for implementing the control functions of a digital computer using programmable control signals in a separate memory called a control store, which is organized on a word basis.

MIR: Memory Information Register in the Interpreter.

Monoprogramming: A general computer operating environment in which one program at a time is executed.

Most Significant Bit (MSB): For a number or value represented in binary notation, that bit position which represents the most significant portion of the number, or the sign of the number.

MPCR: The Microprogram Count Register in the Interpreter.

Multiprogramming: A general computer operating environment in which more than one program at a time is executed with each given a fixed time slice.

Multiprocessing: A general computer operating environment in which two or more central processors execute simultaneously, and share resources; such as, memory and I/O devices.

Multiprocessor: A network of computers capable of simultaneously executing two or more programs or sequences of instructions by means of multiprogramming, parallel processing or both.

Nanoinstruction: A single instruction stored in N-Memory of the Interpreter, the contents of which constitute 56 unique signals for controlling hardware logic of the Interpreter.

Nanomemory (N-Memory): The N-Memory, one of the five functional units of the Interpreter, stores 56 specific enable signals for the hardware logic within the Logic Unit, Control Unit, and Memory Control Unit.

Page: A grouping of addresses of fixed length. In the Emulator the AN/UYK-7 memory was treated as eight pages of 8K each.

PAR: Program Address Register in the AN/UYK-7.

Program Address Register (PAR): A register in the AN/UYK-7 which holds the address of the next instruction to be accessed.

Port Select Unit (PSU): The PSU provides control and the electrical interface between a single Interpreter and its Devices and Data/Program Memory.

Shift Amount Register (SAR): The SAR is a 6-bit register in the Control Unit of the Interpreter and is used to store the number of positions a word or literal is to be shifted by the barrel switch.

Switch Interlock (SWI): The SWI provides the interconnection between Interpreters, Data/Program Memory, and Devices of an Interpreter Based System. Its function is to permit any one of a multiplicity of Interpreters to access all modules of an array of Data/Program Memory and/or all Devices.

Target: A term used in Emulation to define a machine to be emulated (imitated). In this thesis, the AN/UYK-7.

TRANSLANG: A computer programming language designed to convert pseudo-English language statements defining the action of the Interpreter for each machine cycle into binary patterns for the M-Memories.

Vertical: A microprogramming instruction which controls those gates needed for a single function execution.

Z Register Section: A collection of registers and selection gates in the Memory Control Unit of the Interpreter, which includes the CTR, LIT, and Input Selection gates used to control the execution sequence of microinstructions.

BIBLIOGRAPHY

1. Agrawala, A. K. and Rausher, T. G., "Microprogramming: Perspective and Status," IEEE Trans, C23, p. 817-37, August 1974.
2. Allred, G. R., "System/370 Integrated Emulation Under OS and DOS," Proceedings SJCC, 38, p. 163-67, 1971.
3. Almes, G. T., Drongowski, P. J., and Fuller, S. H., "Emulating the Nova on the PDP 11/40: A Case Study," Computer Conference, p. 53-6, Fall 1975.
4. Bagley, J. D., "Microprogrammable Virtual Machines," Computer, p. 38-42, February 1976.
5. Boulaye, G. and Mermet, J., Microprogramming, Herman, Paris, 1972.
6. Bricker, G. B., "Taking the Risk Out of System Upgrading," Data Processing Magazine, 12, p. 27-9, September 1970.
7. Burrough's Corporation Report 64116, Emulation Facility, by Advanced Design Organization, Paoli, Pa., 28 June 1971.
8. Burrough's Corporation Report TR70-2, The Interpreter, by R. L. Davis, 16 February 1970.
9. Burrough's Corporation Report TR70-8, Microprogramming Manual for Interpreter Based Systems, by Advanced Design Organization, Paoli, Pa., November 1970.
10. Burrough's Corporation Report 66143, Algol Reference Manual for the Interpreter Based System, by Advanced Design Organization, Paoli, Pa., 15 June 1975.

11. Dolhoff, T. L., "The Negative Aspects of Microprogramming," *Datamation*, 20, p. 64-6, July 1974.
12. Ellenby, J., "Emulation and Competition in I/O System Design," *Computer Conference*, p. 303-6, 1974.
13. Flynn, M. J., Neuhauser, C., and McClure, R. M., "EMMY-An Emulation System for User Microprogramming," *Proceedings NCC*, 44, p. 85-9, 1975.
14. Galey, J. M., "Microprogramming: The Bridge Between Hardware and Software," *Computer*, p. 23, August 1975.
15. Husson, S. S., *Microprogramming Principles and Practices*, Prentice-Hall, Inc., 1970.
16. Jaeger, R., "Microprogramming: A General Design Tool," *Computer Design*, 13, p. 150-7, August 1974.
17. Jones, L. H., "Instruction Sequencing in Microprogrammed Computers," *Proceedings NCC*, 44, p. 91-8, 1975.
18. Jones, L., "A Survey of Current Work in Microprogramming," *Computer*, p. 33-8, August 1975.
19. Jones, L. H. and Merwin, R. E., "Trends in Microprogramming. A Second Reading," *IEEE Trans*, C23, p. 754-9, August 1974.
20. Kahn, P. G. and Fuller, M. E., "Program Conversion: A Discussion of Techniques," *Data Processing Magazine*, 11, p. 28-31, November 1969.
21. Mallach, E. G., "Emulator Architecture," *Computer*, p. 24-32, August 1975.

22. Mandell, R. L., "Hardware/Software Trade-Offs--Reasons and Directions," Proceedings FJCC, 41, p. 453-9, 1972.
23. Rauscher, T. G., "On the Feasibility of Emulating the AN/UYK-7 Computer on the AADC Signal Processing Element," NTIS, November 1972.
24. Reigel, E. W. V. and Fisher D. A., "The Interpreter--A Microprogrammable Building Block System," Proceedings SJCC, 40, p. 705-23, 1972.
25. Rosin, R. F., "Contemporary Concepts of Microprogramming and Emulation," Computer Survey, 1, p. 197-212, December 1969.
26. Sperry Rand, UNIVAC, Computer Set AN/UYK-7 (V) Technical Manual Volume 1, Navships 0967-319-4010, January 1971.
27. Sperry Rand, UNIVAC, AN/UYK-7, Technical Description.
28. Tucker, A. B. and Flynn, M. J., "Dynamic Microprogramming: Processor Organization and Programming," Communications of the ACM, 14, p. 240-50, April 1971.
29. Wilkes, M. B. "The Growth of Interest in Microprogramming: A Literature Survey," Computer Survey, 1, p. 139-45, September 1969.

INITIAL DISTRIBUTION

| | No. Copies |
|--|------------|
| 1. Defense Documentation Center Cameron Station Alexandria, Virginia, 22314 | 2 |
| 2. Library, Code 0212 Naval Postgraduate School Monterey, California 93940 | 2 |
| 3. Department Chairman, Code 52 Computer Science Group Naval Postgraduate School Monterey, California 93940 | 1 |
| 4. Professor S. Jauregui, Code 62JA (Thesis Advisor) Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940 | 5 |
| 5. LT. Lyle V. Rich, Code 52RS (Second Reader) Department of Computer Science Naval Postgraduate School Monterey, California 93940 | 3 |
| 6. Mr. J. Lynch Burrough's ADO Federal and Special Systems Group P.O. BOX 517 Paoli, Pa. 19301 | 1 |
| 7. Mr Carl Benson Naval Electronics Systems Engineering Center P. O. Box 80337 San Diego, California 92138 | 1 |

8. Mr. J. Lopata 1
Burrough's Corporation
P. O. Box 517
Paoli, Pa. 19301
9. Lt. Jerry M. Haggerty 1
103 Moran Circle
Monterey, California 93940
10. Lt. John M. Hartling 1
376 A. Bergin Drive
Monterey, California 93940
11. Naval Electronics Systems Command 1
Code PME 107
Washington, D. C. 20360
Attn: CAPT W. Flowers
12. Naval Electronics Systems Command 1
Code PME 107
Washington, D. C. 20360
Attn: Mr. R. Matterazza
13. Naval Electronics Systems Command 1
Code PME 107
Washington, D. C. 20360
Attn: CAPT H. Leavitt

Thesis 168148
H1116 Haggerty
c.1 Emulation of the AN/
UYK-7 tactical data com-
puter in the Burrough's
D-machine.

| | |
|-----------|-------|
| 23 MAR 78 | 24877 |
| 6 SEP 78 | 25394 |
| 28 FEB 81 | 26385 |
| 5 DEC 84 | 29982 |

Thesis 100148
H1116 Haggerty
c.1 Emulation of the AN/
UYK-7 tactical data com-
puter in the Burrough's
D-machine.

DUDLEY KNOX LIBRARY



3 2768 00353584 0